博士論文

Machine Learning from Limited Information: Approaches Based on Information Sharing

（限られた情報からの機械学習：情報共有に基づくアプローチ）

山根 一航

Contents

Abstract

In this modern society, due to the highly advanced engineering technologies such as information, communication, sensing, and measurement technologies and the widespread mobile computing environments such as laptops, smart phones, and tablets as well as cloud computing, a wide variety of *data* are being increasingly generated and accumulated every moment, everywhere. Rapid advances of computer architectures and computer science have enabled us to perform more and more various computation tasks, which had never been realized before, in a shorter time, on a larger scale.

Motivated by such demands and opportunities, *machine learning*, the methodology of converting information extracted from data to useful knowledge automatically and efficiently using computers, has been increasing its importance and attracting great attention, being subject to extensive and active studies today.

Machine learning has been achieving remarkable success in a wide range of applications. Its highly flexible problem-solving ability has enabled automation, acceleration, and sophistication on complex tasks that computers had not ever been able to solve satisfactorily. Recent significant milestones in this area include an image recognition system whose accuracy surpassing that of humans and game playing systems beating professional human players on Go and poker.

Such great success of machine learning so far has been relying on the use of high-quality, abundant data. However, it is not always possible to collect a large amount of data with sufficient quality in every application domain. In order to expand the applicability of machine learning, development of methodologies for accurately learning only from *limited information* is of particular importance.

This dissertation discusses machine learning under the situation where training data have only limited information. We consider two types of limitations of information: 1) *quantitative limitation* and 2) *qualitative limitation*. Learning from *quantitatively limited information* refers to the situation where we are required to solve a learning task with a relatively small amount of training data while the learning target could be estimated at a satisfactory level provided that there are a sufficient amount of training data available. On the other hand, learning from *qualitatively limited information* refers to the situation where a learning target cannot be fully identified due to some missing information about it in training data no matter how many training data are given, unless further assumptions are provided.

Quantitative limitation is a commonly encountered issue in various real-world applications of machine learning. We focus on investigating two specific machine learning problems under

this setting and discuss how we can alleviate the issue. The first one is *multi-dimensional log-density gradient estimation*, and the second one is *multi-task principal component analysis*. As we will argue later, *information sharing* approaches based on *multi-task learning* is expected to be effective for these problems.

It seems impossible to perform learning from qualitatively limited information judging from its definition. However, when we have another source of information, the situation may be overcome by incorporating the additional information to fill the missing piece for identification of the target. We show that this is in fact the case in *uplift modeling from separate labels*. We will see that our uplift modeling method can effectively solve the task by *sharing information* between two training data sets obtained from slightly different populations.

In this dissertation, we present approaches based on information sharing to several problems of learning from limited information. More specifically, the main contributions of this dissertation are as follows.

1) We propose a method for multi-dimensional log-density gradient estimation from quantitatively limited information. Our method encourages information sharing between the outputs of the log-density gradient based on a multi-task learning technique regarding each output dimension as a task. In the application of the multi-task learning technique, the design of the way of information sharing is a crucial factor for better performance. In our method, we use models designed based on the general fact that all the output dimensions are derived as partial derivatives of a common primitive function. This enables a distribution-free information sharing method that does not require strong prior knowledge about the task relationship unlike many other multi-task learning methods.

2) We propose a method for solving multiple principal component analysis tasks each of which has only quantitatively limited information for training. In our method, the tasks are solved simultaneously while sharing information among them via a regularizer that makes their solutions close to each other. In principal component analysis, the learning target is a projection matrix, and the space of projection matrices forms a Riemannian manifold whose geometric structure is different from that of the Euclidean space. Hence, the traditional regularizer based on the Euclidean geometry might fail to capture the similarity between projection matrices represented by different bases. Our method uses a regularizer based on the intrinsic geometry of the non-Euclidean manifold, which enables the application of a recently developed optimization technique for directly optimizing the matrices on the manifold. We confirm the usefulness of our method through experiments on synthetic and brain-computer interface data sets.

3) We propose a method for uplift modeling from qualitatively limited information. Roughly speaking, uplift modeling is the problem of analysing the causal relationship between two variables, a treatment variable and an outcome variable. In the standard uplift modeling setup, we are given training samples labeled by both of those two variables, the treatment and the outcome. In our setup, *uplift modeling from separate labels*, we consider qualitatively limited information: We only have one of the two labels for each training

sample, i.e., no training sample has both labels at the same time. It is not in general possible to perform uplift modeling under this setting. However, we show that this problem becomes feasible by *sharing information* between two populations that are slightly different from each other. Furthermore, our method directly estimates the learning target to overcome the instability of a naive approach based on multi-stage estimation. We show the effectiveness of our method through experiments and also give a theoretical guarantee for its performance.

In summary, this dissertation shows the effectiveness of information sharing approaches to learning from limited information. We demonstrate the performance of our methods designed based on those approaches for three problems. The results confirm that information sharing is an effective approach to learning from limited information.

Acknowledgements

First of all, I would like to express my deepest gratitude to my adviser, Prof. Masashi Sugiyama, for his patient and uncompromising guidance for my studies. He went out of his way to offer us great research opportunities and environment to make sure that we can intensively challenge ourselves, while he always gave me a hand when things turned out to be beyond my capability. I sincerely appreciate the constructive and important comments from Prof. Noboru Kunihiro, Prof. Taiji Suzuki, Prof. Yasutoshi Makino, and Prof. Junya Honda on this dissertation. Suggestions from Prof. Taiji Suzuki on theoretical analyses were especially helpful. I am very grateful to Prof. Issei Sato, Prof. Junya Honda, Dr. Gang Niu, Dr. Voot Tangkaratt, and Futoshi Futami for their helpful advice, thoughts, and experiences that they have shared with me. I cannot express my appreciation enough to Prof. Florian Yger and Prof. Hiroaki Sasaki for their great help and suggestions. Many of the key ideas included in this dissertation were conceived out of discussions with them, and it would have been very difficult to complete the dissertation without their help. I would like to make a special mention of Prof. Florian Yger's assistance for my research visits to Paris-Dauphine University. I would like to thank Prof. Jamal Atif and Prof. Maxime Berar for the great collaborations and the valuable discussions. I owe an very important debt to Dr. Marthinus Christoffel du Plessis, Dr. Tomoya Sakai, Takashi Ishida, and Takeshi Teshima. Discussions and conversations with them have been always inspiring and insightful. In particular, I have been influenced greatly by the conversations and thoughts that Dr. Marthinus Christoffel du Plessis has shared with me. I would like to give special thanks to Dr. Tomoya Sakai, Kento Nozawa, Soma Yokoi, and other computer administrators for their efforts to maintain the simulation computers in best conditions. I was able to conduct simulation studies efficiently thanks to them. I would like to show my appreciation to Ms. Etsuko Yoshida, Ms. Yuko Kawashima, and Ms. Kumiko Nakano for advising and helping me with my administrative work as well as to all of my friends and all members of our laboratory for their encouragements. Finally, I deeply thank my parents for their tremendous support, without which I could not have even started my studies.

I was supported by the Japan Society for the Promotion of Science (JSPS) Fellowship Program (the grant number 16J07970) since April, 2016 to March, 2018.

Chapter 1

Introduction

This chapter explains the background, the motivation, the challenges, and the contributions of this thesis. We also give a brief introduction to machine learning and explain issues of learning from limited information and countermeasure approaches based on information sharing.

1.1 Machine Learning

Machine learning has attracted much attention and indeed playing significantly important roles in industry and scientific research today (Friedman et al., 2001; He et al., 2015; Mnih et al., 2013; Simonyan and Zisserman, 2015).

Since it is a central subject of this dissertation, we devote this first section to introduction to machine learning with focus on what machine learning can do, why we study it, and how it has been advanced.

1.1.1 Learning

Learning can be informally defined as any intellectual process of discovering or extracting useful knowledge from concrete examples or observations that contain information about what we are interested in (Murphy, 2012; Shalev-Shwartz and Ben-David, 2014). "Useful knowledge" here includes general laws that explain what have been observed and rules for predicting what will be observed (Friedman et al., 2001; Mohri et al., 2012).

As is suggested by how far humans develop their intelligence from their babyhood to their adulthood, the ability of learning is one of the essential characteristics that make the humanity as intelligent as it is.

1.1.2 Machine Learning

Today, we are facing an exploding amount and variety of data to be analyzed due to the highly developed hardware and software technologies such as mobile computing, cloud computing, telecommunications, and the Internet.

However, data themselves do not serve us much. It is often the case that we do not know how to use, understand, or interpret data in order to draw meaningful knowledge from them

although we do know if they have some information. Data carry information, but it is not knowledge in itself. Data are only a source of knowledge. Data need to be processed in order for their information to turn into knowledge. How can we accomplish this? *Machine learning* has been shown to be a promising answer to this question.

Machine learning is a field of science and engineering for endowing machines (i.e., computers) with the ability of learning from examples and observations fed to them in the form of electronic data (Bishop et al., 2006; Goodfellow et al., 2016; Mitchell, 1997; Mohri et al., 2012; Murphy, 2012; Shalev-Shwartz and Ben-David, 2014). In other words, it aims to make computers *autonomously* extract useful knowledge from data. By doing so, we benefit from their computational power and scalability to process a lot of computational tasks efficiently and automatically. More importantly, computers have a potential to perform learning tasks more accurately than we do because of their precise computation and the capability of handling complex and large data that humans cannot even perceive or memorize. Moreover, machine learning may find new problems that humans have never even attempted to solve.

1.1.3 Machine Learning as Indirect Computer Programming

Let us think a little more about why machine learning should be studied and how it can help us. One of many possible answers is that machine learning provides an easy way of programming computers (Goodfellow et al., 2014; Shalev-Shwartz and Ben-David, 2014).

Since their birth in the middle of the 20th century, general-purpose programmable computers have kept their rapid development (Burks and Burks, 1981). The successive growth of their information processing capability has enabled themselves to perform more and more complex and computationally demanding tasks (Moore, 1998).

However, considerable efforts must be made in order to use computers effectively: We need to program computers to perform what we want them to do. This is not always straightforward if we take a hard-coding strategy. By "hard-coding," we mean programing a computer by telling it exactly what to do step by step. This approach often requires strong expertise about algorithm designs and the task itself to appropriately break down the task into small, simple pieces so that computers can understand how to do without slightest ambiguity. In addition, this approach can lack the capability of adaptation to different tasks. This means that we may need to write as many different programs as the number of tasks.

On the other hand, machine learning does not require us to figure out how exactly for a computer to perform a specific task, but we only have to tell it how to learn to do so from examples (Mohri et al., 2012; Shalev-Shwartz and Ben-David, 2014). An important difference between the two strategies, task-specific hard-coding and programming a computer to learn, is that the latter is a more abstract process. This brings several advantages into the latter strategy:

- We need not necessarily have domain knowledge about the specific task.

- A learning program potentially adapts to many other tasks only by changing input examples.
- Improvement of the learning program implies improvement on all the tasks it solves.

From this viewpoint, machine learning can be seen as another, possibly easy way of programming computers.

Although we have seen the good side of machine learning, it is not a silver bullet; it also has a shortcoming. Machine learning needs examples to feed to a computer. Fortunately, it is often easier for us to show *what* we want to achieve by examples compared to showing *how* to do (Goodfellow et al., 2016; Shalev-Shwartz and Ben-David, 2014). From this viewpoint, machine learning can be seen as an easy way for us to communicate with computers. What is better, we may not even need to produce examples ourselves if there are some alternative data collected or observed by other means, e.g., data available on the Internet.

Although we have motivated machine learning from an engineering viewpoint so far, it is natural to ask questions about the nature of learning process from a more scientific viewpoint: What makes learning possible? How can we make computers learn exactly like humans? What are the differences between human and machine learning? Attempts to answer such questions have been actively made in the neuroscience and the artificial intelligence communities (Hebb et al., 1949; Minsky, 1961). However, the focus of this thesis is more on the engineering side, and we are interested in developing concrete, practical machine learning methods that accurately and efficiently solve real-world problems.

1.2 Learning from Limited Information

The remarkable success of machine learning in the past decades has been built on the use of high quality, ample data (He et al., 2015; Simonyan and Zisserman, 2015). However, it is not always the case that we have access to such rich data both in quality and quantity. Investigation and development of machine learning methods that can learn from *limited information* should be promoted in order to further expand the applicability of machine learning to cover a wider range of real world problems.

With this view in mind, this thesis tackles challenges that appear in *learning from limited information*, where information provided by training data[*1] is limited in several ways.

1.2.1 Two Types of Limitations of Information

We consider the following two types of limitations of information: *quantitative limitation* and *qualitative limitation*.

- *Quantitative limitation:* This corresponds to situations where we are required to perform a learning task with a relatively small number of training samples while the

[*1] *Training* refers to the process of learning as opposed to test evaluation, and *training data* refers to data that we are allowed to use for training.

task can be easily solved provided that we have abundant training samples. Similar situations can happen when training data are so noisy that we need a larger number of samples than those needed in a standard setup in order to achieve a result with comparable quality.

Learning from quantitatively limited information is a ubiquitous challenge that we commonly encounter in various real-world problems. In Chapter 3 and Chapter 4, we will discuss how we can overcome such challenges that arise in *multi-dimensional log-density gradient estimation* and *multi-task principal component analysis* under situations of learning from quantitatively limited information. As will be explained later, *information sharing* approaches base on *multi-task learning* are promising for these specific problems.

- *Qualitative limitation:* When we say training data are qualitatively limited, we mean that training data have some essential deficiency of information about the learning target so that it would be impossible to fully identify the target no matter how many samples are given, without any additional assumptions or prior knowledge provided.

  Learning from qualitatively limited information is infeasible without assumptions. However, it may be feasible when appropriate conditions are given. In fact, as we will see in Chapter 5, *uplift modeling from separate labels* is an instance of learning from qualitatively limited information that we can indeed solve under some reasonable assumptions. An approach based on *information sharing* will play an important role again to design an efficient and effective algorithm for this problem.

1.2.2 Information Sharing Approaches

Even if sufficient information is not provided for a learning task, we may still "borrow" information from other tasks to compensate the deficit. This is the basic idea of *information sharing*. We will see that the issues of limited information described in Section 1.2.1 can be overcome when information can be shared between multiple outputs, learning tasks, or training data sources. More specifically, we will present the following approaches based on information sharing to tackling these issues.

- Information is shared across *multiple outputs* of the target function in learning a *vector-valued function* (Micchelli and Pontil, 2005a) from quantitatively limited information. This is expected to be especially useful when the outputs of the target function are related to each other. *Multi-dimensional log-density gradient estimation* is an important instance of this situation, on which we will investigate more closely in Chapter 3.

- Sharing information across *multiple learning tasks* (Caruana, 1997) can be useful when they are similar to each other but each task has only quantitatively limited information. We investigate *multi-task principal component analysis* from quantitatively limited information as a particular case in Chapter 4.

- We share information across *multiple training data sources* when each data source only has incomplete information, but one can complement their missing information by appropriately combining their information to identify the learning target. This is a situation of learning from qualitatively limited information, which occurs in *uplift modeling from separate labels* studied in Chapter 5.

1.2.3 Information Sharing via Multi-Task Learning

Multi-task learning is a problem setting where we have multiple related learning tasks to solve (Ando and Zhang, 2005; Argyriou et al., 2008a; Baxter, 2000; Caruana, 1998; Evgeniou and Pontil, 2004a; Jacob et al., 2009a; Lozano and Swirszcz, 2012; Obozinski and Taskar, 2006; Thrun, 1996; Zhang, 2013; Zhou et al., 2011). Multi-task learning methods are intended to improve overall learning performance by simultaneously solving the tasks while sharing information between related ones.

This idea can be seen as the analogy of the flexible, and dynamic knowledge transfer observed in human learning in our daily life:

- Students at school often study many subjects such as mathematics, computer science, physics, linguistics, economics, history, and politics, at the same time. These subjects are not totally irrelevant but often very related to each other. Mathematics is a strong tool for computer science, physics, and economics. Linguistics can be related to the historical and the political backgrounds of the place where the language was born and has been developed. By finding connections between multiple subjects, students may be able to acquire better understandings and deep knowledge efficiently.

- Researchers with similar interests often gather as a community to share their ideas, knowledge, techniques, and other research experiences with each other for better understanding their own research topics. Although different researchers seldom work on exactly the same topic, such exchange of information may encourage the progress of each individual work.

Multi-task learning aims to incorporate such a knowledge transfer mechanism into machine learning.

However, there is a critical difference between multi-task learning performed by humans and that performed by computers. The multi-tasking capability of humans may be fairly limited due to the bounded cognitive/physical functions and tolerance to mental burden. Computers, on the other hand, may have much more suitable architectures for multi-tasking in terms of memory capacity, computation speed, and scalability. For this reason, multi-task learning has a great potential for enabling machines to achieve intelligence beyond that humans posses.

Multi-task learning is particularly useful when there are many related learning tasks while each task does not have abundant training data, which is often the case in many real-world problems.

- Take a face authentication system as an example (Gangwar and Joshi, 2016; Menotti et al., 2015; Patel et al., 2016). A face authentication system can be built by learning a classifier for discriminating a user who is supposed to pass the authentication from any other individuals. In an ideal world, such a system could be best calibrated to the specific user by training it on face image data of the user. However, collecting sufficiently many face images from a single user would be time-consuming and can be heavy burden and unpleasant experience for the user.
- Another example is speech recognition (Graves et al., 2013; Hinton et al., 2012). As in the face authentication example, we may want to train a speech recognition system so as to adapt it specifically to the voice of each individual user because different speakers have different voices and accents. Collecting many samples from each single user may be costly, but it may be reasonable to collect a small number of samples for each of many users.

In both examples, separately learning for each individual would suffer from limited information, but we may improve the performance by jointly solving the tasks while sharing information.

1.2.4 Learning Vector-Valued Functions

Many machine learning tasks are aimed at learning real-valued (i.e., scaler-valued) functions. However, we sometimes encounter slightly more complex learning targets, *vector-valued functions* (Micchelli and Pontil, 2005b). Learning a vector-valued function generalizes learning a real-valued function since real numbers form a one-dimensional vector space.

On the other hand, the former can be reduced to the latter by dividing it into the sub-problems of estimating each output dimension of the target vector-valued function. Each of the sub-problems can be solved as a standard real-valued function learning problem if it is solved independently of the others.

However, this does not mean that it is the best way to solve it. When the output dimensions of the vector-valued function are related to each other, it may be possible to estimate them more accurately by jointly learning them in light of multi-task learning (Micchelli and Pontil, 2005b). Treating the sub-problems as independent tasks and separately solving them means giving up on the possibility of improvement and throwing away information that they could share.

As a concrete example, we will demonstrate that *multi-dimensional log-density gradient estimation* can be improved by the information sharing approach in Chapter 3.

1.3 Contributions of This Thesis

This thesis is devoted to investigation of the effectiveness of information sharing approaches to learning from limited information. In particular, we study three types of information sharing approaches to different instances of learning from limited information.

1.3.1 Approaches Based on Information Sharing

Sharing information between output dimensions of a vector-valued function. We investigate the problem of *log-density gradient estimation* (Beran, 1976; Cox, 1985; Sasaki et al., 2014) in multi-dimensional cases, where the *log-density gradient* refers to the derivative of the logarithm of the underlying data density function. When data have multiple dimensions, the log-density gradient will be a vector-valued function.

We propose a method for multi-dimensional log-density gradient estimation based on the idea of multi-task learning (Caruana, 1998). Our proposed method jointly estimates all the dimensions while sharing information across them.

A critical piece of multi-task learning algorithm design in general is how to device the way of information sharing (Ando and Zhang, 2005; Argyriou et al., 2008a; Baxter, 2000; Caruana, 1998; Evgeniou and Pontil, 2004a; Jacob et al., 2009a; Lozano and Swirszcz, 2012; Obozinski and Taskar, 2006; Thrun, 1996; Zhang, 2013; Zhou et al., 2011). Our method uses models reflecting the fact that all output dimensions are obtained by taking partial derivatives of a common function, which generally holds in this problem regardless of the data distribution. This enables information sharing across the tasks in a distribution-free fashion. This result is presented in Chapter 3.

Sharing information based on a non-Euclidean metric. We propose a method for solving multiple tasks of *Principal Component Analysis (PCA)* (Hotelling, 1933; Joliffe, 1986; Pearson, 1901) that are similar to each other, where each task has a limited amount of training data. Our proposed method jointly solves the tasks while sharing information among the tasks via a multi-task learning regularizer that makes solutions closer to each other.

In each PCA task, the learning target is a projection matrix (Hotelling, 1933; Joliffe, 1986; Pearson, 1901). The traditional Euclidean-geometry-based multi-task regularizer might impose unexpected dynamics during the training since the space of projection matrices has a special geometric structure that is different from that of the Euclidean space (Hotelling, 1933; Joliffe, 1986; Pearson, 1901). We propose a multi-task PCA method with a regularizer promoting information sharing designed based on the intrinsic geometry of the space of projection matrices. Our formulation enables the proposed method to enjoy a recent but theoretically well-founded optimization technique (Absil et al., 2009) for efficiently solving the problem. We confirm the usefulness of our method through experiments. We explain the details in Chapter 4.

Information sharing to overcome qualitative limitation of information. We propose a method for *uplift modeling* from qualitatively limited information. Briefly speaking, uplift modeling is the problem of estimating the impact of some variable called a *treatment* on another variable called an *outcome* (Gutierrez and Gérardy, 2017; Jaskowski and Jaroszewicz, 2012; Radcliffe and Surry, 1999, 2011; Rzepakowski and Jaroszewicz, 2012a; Shalit et al., 2017).

In the standard setup of uplift modeling, we assume that each training sample has two types of labels (Gutierrez and Gérardy, 2017; Jaskowski and Jaroszewicz, 2012; Radcliffe and Surry, 1999, 2011; Rzepakowski and Jaroszewicz, 2012a; Shalit et al., 2017): 1) One represents the conducted treatment, and 2) the other represents its outcome.

On the other hand, we consider the situation where every training sample lacks one of the two labels, i.e., no training sample has both labels in which it is not generally possible to perform uplift modeling. We show that uplift modeling becomes feasible even under this situation of qualitatively limited information, if there are two such tasks whose distributions satisfy reasonable assumptions, by sharing information between them.

Furthermore, our method directly produces an estimate of the learning target to overcome the instability of a naive approach based on multi-stage estimation (Imbens, 2014). We show the effectiveness of our method through experiments and also give a theoretical guarantee for its performance. The details are presented in Chapter 5.

The following sub-sections describe our contributions more in detail problem by problem including technical challenges.

1.3.2 Multi-Dimensional Log-Density Gradient Estimation

In Chapter 3, we discuss the problem of *multi-dimensional log-density gradient estimation*. Log-density gradient estimation is the problem of estimating *log-density gradient*, the derivative of the logarithm of the underlying probability density function of given data. It is an important, fundamental problem whose applications include mode-seeking clustering (Fukunaga and Hostetler, 1975; Sasaki et al., 2014), measuring non-Gaussianity of a distribution (Huber, 1985), and other topics in statistics (Singh, 1977).

When data are multi-dimensional, the log-density gradient, is a multi-dimensional vector-valued function. Regarding estimation of its output dimensions (i.e., partial derivatives) as independent tasks, we may apply existing methods (Beran, 1976; Cox, 1985; Sasaki et al., 2014) to separately estimate them.

However, these output dimensions are related to each other in that they are obtained by applying partial derivative operators to a common primitive function, which is the log-density. It is expected that the result of estimation of one output dimension contains information that is useful for estimation of other dimensions.

In this research, we propose a multi-task approach to multi-dimensional log-density gradient estimation that simultaneously estimates all the output dimensions while sharing information across them regarding estimation of each output dimension as a task.

More specifically, we design a regularizer that encourages information sharing among the tasks with no strong assumption required on the task relationships in a distribution-free fashion. What makes it possible is that we design models based on the general fact that all output dimensions are partial derivatives of a common primitive function.

We demonstrate that our proposed method is able to estimate the log-density gradient accurately through experiments on synthetic and real data sets. We also show that a

mode-seeking clustering method based on the proposed estimator performs well on clustering tasks on synthetic and real data sets.

1.3.3 Multi-Task Principal Component Analysis

Principal Component Analysis (PCA) is a popular method for unsupervised linear dimensionality reduction (Hotelling, 1933; Joliffe, 1986; Pearson, 1901). It is used for visualization, feature extraction for classification and regression, and other pre-processing purposes. PCA tries to find the optimal orthogonal projection of a fixed rank with which projected data points have the largest variance so that the they will have small overlaps with each other and different points can be easily discriminated.

In some real-world applications of PCA such as *Brain Computer Interfaces (BCI)* (Lotte et al., 2007; Yger et al., 2015), we are only given a *limited amount* of training data due to expensive data collection cost. When we have multiple, similar tasks for each of which we only have a limited amount of data, this situation can be alleviated by sharing information between them within the multi-task learning framework (Caruana, 1998). Here, by the similarity of tasks, we mean that the types of their target functions and training data are the same while they have independent observations of data following similar but possibly different data distributions. To take the heterogeneity into account, sharing models in a soft manner, i.e. making learning results close to each other rather than sharing a common model among the tasks, is expected to be effective (Evgeniou et al., 2005; Evgeniou and Pontil, 2004b).

A challenge here is that the learning target of each PCA task is a projection matrix, and projection matrices form a non-Euclidean manifold (Absil et al., 2009). The traditional soft model-sharing scheme based on the Euclidean geometry might impose unnatural dynamics in the training of the models or fail to correctly measure the similarity between projection matrices. Moreover, the manifold of the projection matrices has a much lower dimensionality than that of their superficial matrix representations (Absil et al., 2009). Naive treatment of those matrices in the Euclidean space ignoring their structure ends up working with an unnecessarily higher-dimensional problem.

To mitigate this issue, we propose a multi-task PCA method based on the metric defined on the manifold of the projection matrices, allowing direct optimization on the manifold by applying a recently developed, theoretically well-founded optimization technique for efficiently solving the problem (Absil et al., 2009).

We demonstrate the effectiveness of our method through synthetic and BCI data sets. This work is presented in Chapter 4.

1.3.4 Uplift Modeling from Separate Labels

In Chapter 5, we consider *uplift modeling from separate labels*. In uplift modeling, a central task is to estimate the impact of a treatment (e.g., medical treatment, an advertisement

campaign) on the change of its outcome (e.g., the rate of recovery from a disease, the amount of purchases). We call the impact of a treatment the *treatment effect*.

In many real-world applications, we are particularly interested in knowing for what kind of individual a treatment is effective and how much it is so in order to decide whether to give the treatment on the individual level. In such cases, it is useful to estimate the *individual treatment effect (ITE)*, the average treatment effect conditioned on the features of an individual (Shalit et al., 2017). We refer to this estimation problem as uplift modeling in what follows.

Conventional methods of uplift modeling require every sample of an individual to be jointly equipped with two types of labels: 1) the treatment given to the individual and 2) its outcome (Gutierrez and Gérardy, 2017; Jaskowski and Jaroszewicz, 2012; Radcliffe and Surry, 1999, 2011; Rzepakowski and Jaroszewicz, 2012a; Shalit et al., 2017). However, obtaining these two labels for each instance at the same time is difficult or expensive in some cases. For example, suppose that we want to know the effect of an E-mail advertisement. We would know to whom we deliver the advertisement as a sender, but it would be difficult to know how much purchases each recipient eventually makes unless we track the individual with a malware or by other unethical means. On the other hand, we may know the amount of purchases when they are actually made. In this case, however, it is often difficult to precisely know whether the purchasers have received the advertisement.

In Chapter 5, we consider a more practical setting of uplift modeling (Gutierrez and Gérardy, 2017; Jaskowski and Jaroszewicz, 2012; Radcliffe and Surry, 1999, 2011; Rzepakowski and Jaroszewicz, 2012a; Shalit et al., 2017), called *uplift modeling from separate labels*, where only one of the two types of labels is available for each sample.

Estimation of the ITE in this setting is not in general feasible when training data are collected from a single population. However, we will show that it becomes possible when we have two kinds of training data sampled from two different populations under some reasonable assumptions.

In this case, the ITE is characterized by four functions each of which can be estimated by simply performing regression on a relevant subset of the training data set. However, a naive approach of separately estimating these four functions and combining them to construct an estimate of the ITE in a post-processing manner is often unstable, Furthermore, improving on the regression tasks does not guarantees a higher-quality estimate of the ITE.

We propose a method that avoids the intermediate sub-tasks of learning the four functions but directly estimates the target function. Our proposed method uses all training data to estimate the target function at once unlike the naive method. This can be seen as another form of information sharing, and it is expected to bring performance improvement to the proposed method.

We show the effectiveness of the proposed approach theoretically and empirically. This topic is covered in Chapter 5.

1.4 Organization of this Dissertation

This dissertation is organized as follows. In Chapter 2, we briefly introduce basic notions and overview some machine learning problems related to our work. From Chapter 3 to Chapter 5, we present our work with detailed explanations about the motivation, our approaches, and experiments, subject by subject. In Chapter 3, we introduce multi-dimensional log-density gradient estimation and our method with application to clustering. Chapter 4 focuses on multi-task principal component analysis and proposes our method for this problem. Chapter 5 is on the problem of uplift modeling from separate labels. We present our method for accurately and efficiently solving it. Finally, we conclude our work with some future prospectives of this line of research in Chapter 6.

Chapter 2

Basics of Machine Learning

In this chapter, we introduce basics of machine learning in order to give an overview of topics related to our work in a broader scope of the field. We also define some notion, terminology, and notation that will be important in the subsequent chapters.

2.1 Notation and Assumptions

$\mathbb{R}$ denotes the set of all real numbers. $\mathbb{N}_+$ denotes the set of all positive natural numbers: $\mathbb{N}_+ := \{1, 2, \dots\}$. For any $n \in \mathbb{N}_+$, $[n] := \{1, \dots, n\}$.

When it helps avoid confusion, we use different fonts and styles for mathematical symbols depending on what they represent as follows. Vectors and vector-valued random variables are denoted by lowercase, bold Roman letters (e.g., $\boldsymbol{x}$, $\boldsymbol{\alpha}$). For matrices and matrix-valued random variables, we use uppercase, bold Roman letters (e.g., $\boldsymbol{X}$, $\boldsymbol{U}$). We use underlined letters for realizations of random variables and constants (e.g., $\underline{\boldsymbol{x}}$, $\underline{y}$). Sets are denoted by uppercase calligraphic letters (e.g., $\mathcal{X}$, $\mathcal{Y}$). However, we do not necessarily follow these rules strictly and there may be some exceptions.

For any random variables $x_1, \dots, x_n$ with $n \in \mathbb{N}_+$, we define the following symbols. $D_{x_1,\dots,x_n}$ denotes their joint probability distribution. For any $m \in [n]$, $D_{x_1,\dots,x_m|x_{m+1},\dots,x_n}$ denotes the conditional distribution of $(x_1, \dots, x_m)$ conditioned on $(x_{m+1}, \dots, x_n)$. $\mathbf{E}_D[\cdot]$ denotes the expectation of the variable in the argument, and $\Pr_D[\cdot]$ denotes the probability of the statement in the argument, over the distribution D. We may omit those subscripts when it is clear what are omitted from the context.

We assume that the probability density function exists for any random variable whenever the following symbols are used. $p_{x_1,\dots,x_n}$ denotes joint probability density function (if they are continuous) or their joint probability mass function (if they are discrete). For any $m \in [n]$, $p_{x_1,\dots,x_m|x_{m+1},\dots,x_n}$ denotes the conditional probability density/mass function of $(x_1, \dots, x_m)$ conditioned on $(x_{m+1}, \dots, x_n)$.

For any matrix $\boldsymbol{M}$, $\mathrm{rank}(\boldsymbol{M})$ denotes the rank of $\boldsymbol{M}$ and $\boldsymbol{M}^\top$ denotes its transpose.

2.2 Machine Learning as Risk Minimization

In this section, we introduce a general way of formalizing the goal of a machine learning problem in terms of mathematical optimization and the widely employed learning framework, *empirical risk minimization* (Vapnik, 1995).

2.2.1 Basic Terminologies

Suppose that we want to predict the value of a $\mathcal{Y}$-valued random variable y that (possibly) depends on another $\mathcal{X}$-valued random variable x, where $\mathcal{X}$ and $\mathcal{Y}$ are some measurable spaces. We call y the *output variable* and x the *input variable* as the task is essentially about learning the input-output relationship between these variables.

Machine learning is aimed at solving such a prediction task, but a machine learning algorithm itself does not directly make predictions. Instead, by processing available information from data, it produces another function mapping from $\mathcal{X}$ to $\mathcal{Y}$ whose outputs correspond to predictions. Such a function used for prediction is called a *hypothesis function*, and the set of possible hypothesis functions a learning algorithm may produce is called its *hypothesis class*. Data that a learning algorithm is allowed to use in order to produce a hypothesis, are called *training data*, and data used for test evaluation are called *test data*.

2.2.2 Formulation with Mathematical Optimization

Many machine learning problems are formulated in the form of mathematical optimization problems, i.e., problems of finding a hypothesis function that minimizes/maximizes some objective functional under some constraints. There are several advantages with this approach.

- We can explicitly and objectively quantify the goodness of hypotheses by the objective functional of the optimization problem. This also enables us to mathematically write down the learning target as the optimal solution to the optimization problem even when it is not explicitly defined.
- We can ensure conditions that hypotheses must satisfy by specifying them as constraints. This is more reliable than enforcing the conditions by post-processing a hypothesis function in an ad-hoc fashion since it will be guaranteed that the optimal solution will be not worse than any other eligible ones in terms of the objective.
- Statistical properties of a learning algorithm can be studied in a rigorous way by analyzing the behaviours of an optimization problem and its solutions.
- A lot of useful results from the optimization theory can be readily employed to design efficient learning algorithms.
- Algorithm designs can be broken down into statistical and computational aspects in a modular way. The optimization problem can be designed so that the resulting statistical properties of the solution will be what we want without paying too much

attention to computational issues. Once the optimization problem is defined, we only have to find an optimization algorithm to solve it.[*1]

Below, we will briefly explain some more details about this general approach to machine learning and various examples of concrete methods based on risk minimization.

2.2.3 Loss Function and Risk

Design of a learning algorithm can be divided into two factors: choosing a hypothesis class and establishing a way to find a good hypothesis from the hypothesis class with the help of training data.

For finding a "good" hypothesis, we need to define what is good and what is bad. For this purpose, we use a *loss function*. A *loss function* is a function for measuring the badness[*2] of a hypothesis function in a point-wise manner: It evaluates the performance of a hypothesis function at any data point in terms of how poorly it predicts or explains the data point. We refer to an output of a loss function as the *loss*. In a very general form, a loss function $\widetilde{\ell}$ can have the following form: $\widetilde{\ell} : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. Then, the loss of a hypothesis h at any data point $(x, y) \in \mathcal{X} \times \mathcal{Y}$ can be measured by $\widetilde{\ell}(h, x, y) \in \mathbb{R}$. The smaller $\widetilde{\ell}(h, x, y)$ is, the better h is.

Now, we can assesses a hypothesis *locally* at each point by a loss function, but how can we evaluate its *overall* performance? This can be done by averaging the loss evaluations over all data points. More formally, we define the *risk* of h associated with $\widetilde{\ell}$ as the expected loss:

$$R_{\widetilde{\ell}}(h) := \mathop{\mathbf{E}}_{(x,y)\sim D}[\widetilde{\ell}(h, x, y)], \tag{2.1}$$

where D is the underlying joint distribution of (x, y), and we assume that the expectation exists and ranges in $(-\infty, \infty]$.

Remark 2.2.1. We may use different loss functions for training and test evaluation. A reason is that the type of data may be different between the training and the test phases in *unsupervised learning* and *weakly supervised learning*. Another reason is a performance concerning issue. Training tends to involve heavy computation and a lot of repeated evaluations of a loss function for optimization while it suffices to evaluate the loss only once at each test data point for testing. Approximations or other alternatives to the original loss function are often used for efficient training (Bartlett et al., 2006).

Remark 2.2.2. As we will introduce later, learning problems can be roughly categorized into (at least) two types depending on the form of data points. One is *supervised* learning, where samples of the output variable y are given along with samples of the input variable x both in training data. The other is *unsupervised* learning, where output samples are not available in the training phase, or even in the test evaluation phase. In the supervised case,

[*1] In some cases, we need to make a non-trivial modification to the optimization problem itself to make it feasible. For instance, we often make an approximation to the objective functional used in classification (Bartlett et al., 2006).

[*2] The negative of the output of a loss function defines the goodness of a hypothesis.

we usually use a simpler form of a loss function: $\widetilde{\ell}(h, x, y) := \ell(h(x), y)$ with some function $\ell : \mathcal{Y}^2 \to \mathbb{R}$. Note that ℓ only evaluates h through its outputs. On the other hand, in the unsupervised case where we cannot access output samples, the loss function should not use output samples either, and thus a training loss function typically takes another simpler form as follows: $\widetilde{\ell}(h, x, y) := \ell(h, x)$ with some function $\ell : \mathcal{H} \times \mathcal{X} \to \mathbb{R}$.

2.3 Supervised Learning

The goal of *supervised learning*, in general, is to learn a function for accurately predicting the value of the output variable y given the input variable x under *direct supervision*. Here, the direct supervision refers to the presence of samples of the output variable y labeling each instance x which directly tells the (noisy) ground truth of the prediction task.

2.3.1 Performance Evaluation

In a typical supervised learning task, the loss function ℓ takes any $y \in \mathcal{Y}$ and the prediction of a hypothesis h at x as input, and outputs the loss $\ell(h(x), y)$ that measures their discrepancy, or the prediction error of h at x against y. The goal is to find a hypothesis that minimizes the following risk:

$$R_\ell(h) := \mathop{\mathbf{E}}_{(x,y)\sim D}[\ell(h(x), y)], \tag{2.2}$$

where D is the underlying joint distribution of x and y.

For test evaluation in practice, D is often unknown, and we cannot access the exact value of the expectation in Eq. (2.2). However, it can be approximated by the following sample average using test data:

$$\widehat{R}(h; \mathcal{S}_{\text{te}}) := \frac{1}{n'} \sum_{i \in [n']} \ell(h(x'_i), y'_i), \tag{2.3}$$

where $\mathcal{S}_{\text{te}} := \{(x'_i, y'_i)\}_{i=1}^{n'}$ is a set of test data consisting of samples of x and y, independently and identically distributed (i.i.d.) by D.

2.3.2 Training Data in Supervised Learning

Supervised learning problems are described as *supervised* because there is direct supervision about the output variable provided to the learning algorithm through training data. More specifically, in supervised learning, a training dataset is a set of i.i.d. sample pairs of x and y: $\{(x_i, y_i)\}_{i=1}^{n} \overset{\text{i.i.d.}}{\sim} D$, where D is the same joint distribution as that of the test data. Each instance x_i is annotated by its corresponding output y_i, directly telling us what can be a potential value of y when $x = x_i$.[*3]

[*3] The observation y_i might not be the best prediction for x_i unless y is not deterministically dependent on x. This is a reason why memorizing training data and using their exact values as predictions may not be optimal. Statistical inference techniques are used to deal with this uncertainty.

2.3.3 Classification

Classification is a typical supervised learning problem, whose goal is to learn a function assigning a categorical label y to any test data instance x. By y being *categorical*, we mean that the set of its possible values $\mathcal{Y}$ is a finite set without algebraic, geometric, or any other kind of structure presumably equipped. In classification, elements of $\mathcal{Y}$ are often called *classes* or *class labels*.

We usually use a loss function $\ell_{\text{cls}} : \mathcal{Y} \times \mathcal{Y}$ defined by $\ell_{\text{cls}}(y, y') = 1[y \neq y']$, which measures the disagreement of the two arguments. The risk of a hypothesis h associated with ℓ_{cls} is expressed as

$$R_{\text{cls}}(h) := \mathop{\mathbf{E}}_{D}[\ell_{\text{cls}}(h(x), y)] = \mathop{\mathbf{E}}_{D}[1[y \neq y']] = \Pr[y \neq y'].$$

We call it the *classification risk* or *classification error rate*.

Surprisingly, in spite of their discrete nature, it has been shown that classification problems can be *approximately* solved *with high probability* via continuous optimization with remarkable success (Bartlett et al., 2006; Friedman et al., 2001; He et al., 2015; Simonyan and Zisserman, 2015; Vapnik, 1995).

This is one of the areas where machine learning methods can perform as well as or even better than humans do (He et al., 2015).

2.3.3.1 Bayes Optimal Classifier

Once we have the criterion for performance evaluation defined by the classification risk R_{cls}, we can define a best possible hypothesis h^* as one that achieves the smallest risk: $h^* \in \inf_{h \in \mathcal{F}_{\mathcal{X},\mathcal{Y}}} R_{\text{cls}}(h) = \inf_{h \in \mathcal{F}_{\mathcal{X},\mathcal{Y}}} \Pr[h(x) \neq y]$, where $\mathcal{F}_{\mathcal{X},\mathcal{Y}}$ is the set of all measurable functions mapping from $\mathcal{X}$ to $\mathcal{Y}$. Any hypothesis of $\mathcal{F}_{\mathcal{X},\mathcal{Y}}$ that achieves $\text{BErr}_{\tilde{\ell}}$ is called a *Bayes optimal* classifier.

2.3.3.2 Binary Classification

Binary classification is a special case of classification where only two classes exist. For notational convenience, we encode the two classes by labels: $\mathcal{Y} := \{-1, 1\}$,[*4] and we allow hypothesis functions to output real values, whose signs predict class labels. In the standard binary classification, we consider the zero-one loss function $\ell_{0\text{-}1}$ as the loss function, defined by

$$\ell_{0\text{-}1}(z, y) := 1[\text{sign}(z) \neq y] = 1[z \cdot y \geq 0],$$

for any $z \in \mathbb{R}$ and any $y \in \{-1, 1\}$, where $1[\cdot]$ is the indicator function that outputs 1 if its argument is true and 0 otherwise, and $\text{sign}[\cdot]$ is 1 if the sign of its argument is non-negative, and -1 otherwise. When z is the output of a hypothesis h on x, and y is the corresponding

[*4] Note that this is always possible and does not lose generality.

true class label, the zero-one loss $\ell_{0\text{-}1}(z, y) = \ell_{0\text{-}1}(h(x), y)$ penalizes the hypothesis h by unit loss 1 if and only if its prediction is different from the true class label y.

The risk in this case is referred to as the *zero-one risk* and given by

$$R_{0\text{-}1}(h) = \mathop{\mathbf{E}}_{(x,y)\sim D}[\ell_{0\text{-}1}(h(x), y)] = \mathop{\mathbf{E}}_{(x,y)\sim D}[1[y \cdot h(x) \geq 0]] = \Pr_{(x,y)\sim D}[y \cdot h(x) \geq 0].$$

We can see that the risk coincides with the probability of h making a wrong prediction. It can be shown that a hypothesis h minimizes the zero-one risk if

$$\text{sign}[h(x)] = \text{sign}[p(y = 1 \mid x) - p(y = -1 \mid x)],$$

where $p(y \mid x)$ is the conditional probability mass function.

Since the distribution D is usually unknown and the risk above cannot be exactly calculated, we use the following sample average on the training data:

$$\widehat{R}_{0\text{-}1}(h; \mathcal{S}_{\text{tr}}) := \frac{1}{n} \sum_{i \in [n]} \ell(h(x_i), y_i). \tag{2.4}$$

We refer to the approximated risk as the *empirical risk*, and the exact risk as the *population risk* as opposed to the empirical risk. This framework of learning a function by minimizing the empirical risk is called *empirical risk minimization* (Vapnik, 1995).

Minimizing $\widehat{R}_{0\text{-}1}(h; \mathcal{S}_{\text{tr}})$ involves a discrete objective function, making the optimization hard. In fact, it was proven that there is a simple problem instance where it is NP-hard to find a hypothesis function that makes the empirical zero-one risk greater than $1/2$ (Feldman et al., 2012).

Fortunately, what we want to minimize is not the empirical risk but the population risk, and the exact minimization of the empirical risk is not much of our interest here since $\widehat{R}(h; \mathcal{S}_{\text{tr}})$ is already an approximation to the population risk. We use a continuous approximation to the zero-one loss function to overcome the computational issue. Such approximate loss functions used for this purpose are called *surrogate loss functions*. This approximation is justified as long as the learning result yields a consistent estimator to the minimizer of the population risk, and many surrogate loss functions are guaranteed satisfy this property (Bartlett et al., 2006). By introducing surrogate loss functions, continuous optimization techniques such as gradient-based methods will be applicable to solve the problem. Several surrogate loss functions have been proposed in the literature depending on the purposes. We will give a few examples below.

- Logistic loss: $\ell_{\text{logi}}(z, y) := -\log(1 + \exp(-z \cdot y))$. This is a convex upper-bound of the zero-one loss. The logistic loss of a hypothesis h, $\ell_{\text{logi}}(h(x), y)$, can be seen as the log-likelihood of the following probabilistic model for the conditional probability

density $p(y \mid x)$:

$$q(y \mid x; h) := \frac{\exp(y \cdot h(x)/2)}{\exp(h(x)/2) + \exp(-h(x)/2)} = \frac{1}{1 + \exp(-y \cdot h(x))}$$

An intuitive explanation of this model is that h controls the magnitude of the probability through its exponential as a proxy so that the output will be always non-negative. Then, we normalize it over y to make sure that the resulting output values sum up to 1.

Since the logistic loss function is differentiable, the empirical risk minimization can be performed by gradient-based methods. Furthermore, for linear-in-parameter models, the problem will be convex and the global minimizer can be obtained efficiently by convex optimization solvers.

- Hinge loss: $\ell_{\text{hinge}}(z, y) := \max\{1 - z \cdot y, 0\}$. It is another convex upper-bound of the zero-one loss. The resulting empirical risk minimization with a ℓ_2-regularizer corresponds to the soft-margin support vector machines (Vapnik, 1995).

  The hinge loss function is sub-differentiable, and the empirical risk can be minimized performed by sub-gradient-based methods. It is also convex, and the problem can be solved by convex optimization methods for linear-in-parameter models.

- Squared loss: $\ell_{\text{sq}}(z, y) := (1 - z \cdot y)^2$. The squared loss function is also convex and upper-bounds the zero-one loss function. Another expression of this function is $\ell_{\text{sq}}(z, y) = (z - y)^2$, where we can see that the loss encourages the prediction z to be close to the true label y. We can obtain an analytic solution when a linear-in-parameter model is used.

- Sigmoid loss: $\ell_{\text{sig}}(z, y) := 1/[1 + \exp(-z \cdot y))]$. This is a differentiable approximations to the zero-one loss which has relatively less approximation errors compared to the loss functions above. It is differentiable at every point, and we can minimize the corresponding empirical risk by the gradient descent approach as long as the model is also differentiable. On drawback is that the problem will be non-convex even for linear-in-parameter models.

Remark 2.3.1. Note that a choice of the surrogate loss function is a part of a learning algorithm design, not of the problem. Hence, a surrogate loss function should be used only for training purposes but not for evaluation purposes. Once we decided to use the zero-one loss function as the performance measure, we should use it in the ultimate performance evaluation. Otherwise, the evaluation will be unfairly in favor of the algorithm.

2.3.3.3 Multi-Class Classification

A more general setting where there are more than one classes (i.e., $|\mathcal{Y}| \geq 2$) as opposed to binary classification is called *multi-class classification*. Without loss of generality, assume that $\mathcal{Y} = \{1, \ldots, K\}$, where $K \in \mathbb{N}_+$.

Although as a problem setting, it is a natural extension of binary classification, surrogate loss functions used for binary classification may not be easily generalized to the multi-class case since the classes can no longer be simply encoded as signs of real numbers. We only introduce the soft-max cross entropy loss since it is a poplar multi-class surrogate loss function although there are other useful loss functions.

Soft-Max Cross Entropy Loss: We suppose that the hypothesis class $\mathcal{H}$ is a subset of the following set of functions $\{\boldsymbol{h} : \mathcal{X} \to \mathbb{R}^K\}$, and that a hypothesis $\boldsymbol{h} \in \mathcal{H}$ predicts the output y on x by $\operatorname{argmax}_{k \in [K]} h_k(x)$,[*5] where $h_k(x)$ is the k-th element of the vector $\boldsymbol{h}(x)$.

The *soft-max cross entropy loss* $\ell_{\mathrm{MSCE}} : \mathbb{R}^K \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a multi-class extension of the logistic loss, defined by

$$\ell_{\mathrm{MSCE}}(\boldsymbol{h}, y, x) := -\log \left(\frac{\exp\left(h_y(x)\right)}{\sum_{k \in [K]} \exp\left(h_k(x)\right)} \right).$$

This can be seen as the negative log-likelihood of the following probabilistic model for the conditional probability density $p(y \mid x)$:

$$q(y \mid x; \boldsymbol{h}) := \frac{\exp\left(h_y(x)\right)}{\sum_{k \in [K]} \exp\left(h_k(x)\right)}.$$

The logistic loss is the special case of the soft-max cross entropy loss when $k = 2$ and the hypothesis class $\mathcal{H}$ is restricted by $h_2 = -h_1$ for every $\boldsymbol{h} \in \mathcal{H}$.

2.3.4 Regression

Regression refers to estimation of a function for predicting a real-valued output variable y given an input variable $\boldsymbol{x}$. In our notation, $\mathcal{Y} = \mathbb{R}$. There are several loss functions for regression including the following ones.

- Squared loss function: $\ell_{\mathrm{sq}}(z, y) := (z - y)^2$.[*6] In particular, the empirical risk minimization under the squared loss function and a linear model is called *least-squares*, and *ridge regression* when solved with the ℓ_2-regularizer. When we allow a hypothesis to be any function in L_2, a minimizer of the risk with this loss function is the conditional expectation of y conditioned on $\boldsymbol{x}$: $\mathbf{E}[y \mid \boldsymbol{x}] \in \operatorname{arginf}_{f \in L_2} \mathbf{E}[\ell_{\mathrm{sq}}(f(\boldsymbol{x}), y)]$.
- ℓ_1-loss function: $\ell_1(z, y) := |z - y|$. The squared loss function is sensitive to outliers, i.e., abnormal data points distant from normal ones, since errors on those points are

[*5] We take the smallest one if the maximizer is not unique.

[*6] This coincides with the squared loss function for binary classification when the second argument is restricted to $\{-1, 1\}$.

penalized by large loss that quadratically increases with respect to the errors. As a more robust loss function, we may use ℓ_1-loss defined above. This loss only penalize errors linearly, and the result will be less affected by outliers.

2.4 Unsupervised Learning

In *unsupervised learning* problems, output labels are absent in training data unlike supervised ones. In some cases, they are not available even in test data or not defined.

We will give a few examples that will be also important in the subsequent chapters.

2.4.1 Data and Distribution

Typically, a training dataset in an unsupervised problem only consists of input samples: $\mathcal{S}_{\mathrm{tr}} := \{x_i\}_{i=1}^n \overset{\text{i.i.d.}}{\sim} D$. Unsupervised problem are called *unsupervised* because direct supervision about the output is absent as opposed to supervised learning.

2.4.2 Principal Component Analysis

Principal component analysis (PCA) (Hotelling, 1933; Joliffe, 1986; Pearson, 1901) is a widely used unsupervised method to reduce data dimensionality for visualization and data preprocessing purposes.

PCA finds an orthogonal projection of data points onto a lower-dimensional linear subspace on which the data points have the largest variance, hoping that the data points will be widely spread after the projection and easily distinguished from each other.

Suppose that we have i.i.d. $\mathbb{R}^d$-valued training data $\{\boldsymbol{x}_i\}_{i=1}^n$ following some distribution $D_{\boldsymbol{x}}$. We assume that the data have mean zero. If this does not hold, we can *center* them, i.e., subtract the sample mean from the data: $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \overline{\boldsymbol{x}}$ for every $i \in [n]$, where $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i\in[n]} \boldsymbol{x}_i$. Let $\mathcal{P}$ denote the set of orthogonal projection matrices of rank k on $\mathbb{R}^d$: $\mathcal{P} := \{\boldsymbol{P} : \mathbb{R}^d \to \mathbb{R}^d \mid \boldsymbol{P}^2 = \boldsymbol{P}, \boldsymbol{P}^\top = \boldsymbol{P}, \mathrm{rank}(\boldsymbol{P}) = k\}$, where $\mathrm{rank}(\cdot)$ is the rank of the matrix in the argument. Note that $\boldsymbol{P}^2 = \boldsymbol{P}$ is the condition for $\boldsymbol{P}$ to be a projection, and $\boldsymbol{P}^\top = \boldsymbol{P}$ is for its orthogonality. We define the *reconstruction error* of $\boldsymbol{P} \in \mathcal{P}$ on a point $\boldsymbol{x}$ as $\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}$. Then, PCA seeks the orthogonal projection matrix that minimizes the mean squared reconstruction error as follows:

$$\operatorname*{argmin}_{\boldsymbol{P}\in\mathcal{P}} \mathbf{E}[\|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|_2^2], \tag{2.5}$$

where $\|\cdot\|_2$ is the ℓ_2 norm, and the expectation is taken over $\boldsymbol{x} \sim P$.

Using the conditions $\boldsymbol{P}^2 = \boldsymbol{P}$ and $\boldsymbol{P}^\top = \boldsymbol{P}$, we have

$$\begin{aligned}
\mathbf{E}\left[\|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|_2^2\right] &= \mathbf{E}\left[\|\boldsymbol{x}\|_2^2 - \boldsymbol{x}^\top\boldsymbol{P}^\top\boldsymbol{x} - \boldsymbol{x}^\top\boldsymbol{P}\boldsymbol{x} + \boldsymbol{x}^\top\boldsymbol{P}^\top\boldsymbol{P}\boldsymbol{x}\right] \\
&= \mathbf{E}\left[\|\boldsymbol{x}\|_2^2 - \|\boldsymbol{P}\boldsymbol{x}\|_2^2\right] \\
&= \mathrm{Var}[\boldsymbol{x}] - \mathrm{Var}[\boldsymbol{P}\boldsymbol{x}],
\end{aligned}$$

where Var[·] denotes the covariance matrix of the variable in the argument, and the last equation follows since $\boldsymbol{x}$ has mean zero. Note that Var[$\boldsymbol{x}$] is a constant with respect to $\boldsymbol{P}$ and irrelevant to the optimization. Hence, Eq. (2.5) is equivalent to maximizing the variance of the projected data $\boldsymbol{Px}$. Since $\mathrm{Var}[\boldsymbol{Px}] = \mathbf{E}[\mathrm{Tr}(\boldsymbol{x}^\top \boldsymbol{P}^\top \boldsymbol{Px})] = \mathbf{E}[\boldsymbol{P}\,\mathrm{Tr}(\boldsymbol{xx}^\top)] = \mathrm{Tr}(\boldsymbol{P}\,\mathrm{Var}[\boldsymbol{x}])$, Eq. (2.5) is equivalent to

$$\operatorname*{argmax}_{\boldsymbol{P}\in\Pi} \mathrm{Tr}(\boldsymbol{PC}), \tag{2.6}$$

where $\boldsymbol{C} = \mathrm{Var}[\boldsymbol{x}]$.

If the dimensionality of the projected space is k, the projection matrix $\boldsymbol{P}$ can be written as $\boldsymbol{P} = \boldsymbol{U}^\top \boldsymbol{U}$, where $\boldsymbol{U} \in \mathbb{R}^{d\times k}$ is an skinny orthonormal matrix that satisfies $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k$. Hence, our problem can be expressed as

$$\operatorname*{argmax}_{\boldsymbol{U}\in\mathbb{R}^{d\times k}} \mathrm{Tr}(\boldsymbol{UU}^\top \boldsymbol{C}) \text{ s.t. } \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k.$$

All the stationary solutions of the problem satisfy the following *Karush-Kuhn-Tucker (KKT)* conditions:

$$\nabla_{\boldsymbol{U}} \mathcal{L}(\boldsymbol{U}, \boldsymbol{\Lambda}) = \boldsymbol{0}, \quad \nabla_{\boldsymbol{\Lambda}} \mathcal{L}(\boldsymbol{U}, \boldsymbol{\Lambda}) = \boldsymbol{0}, \tag{2.7}$$

where $\mathcal{L}(\boldsymbol{U}, \boldsymbol{\Lambda}) := \mathrm{Tr}(\boldsymbol{UU}^\top \boldsymbol{C}) - \mathrm{Tr}(\boldsymbol{\Lambda}(\boldsymbol{U}^\top \boldsymbol{U} - \boldsymbol{I}))$ is the Lagrangian of the problems, and $\boldsymbol{\Lambda} \in \mathbb{R}^{k\times k}$ is the matrix consisting of the Lagrange multipliers. Equivalently, we have

$$\boldsymbol{CU} = \boldsymbol{U\Lambda}, \quad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k. \tag{2.8}$$

We observe $\boldsymbol{\Lambda} = \boldsymbol{U}^\top \boldsymbol{CU}$. Since $\boldsymbol{C}$ is symmetric and positive semi-definite, so is $\boldsymbol{\Lambda}$. Hence, it has an eigenvalue decomposition: $\boldsymbol{\Lambda} = \widetilde{\boldsymbol{U}}^\top \boldsymbol{\Sigma} \widetilde{\boldsymbol{U}}$, where $\widetilde{\boldsymbol{U}} \in \mathbb{R}^{d\times k}$ is a skinny orthogonal matrix, and $\boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}$ is a diagonal matrix whose elements are all non-negative. Let $\boldsymbol{V} := \boldsymbol{U}\widetilde{\boldsymbol{U}}$. Then, we can see that (2.8) can be equivalently expressed as

$$\boldsymbol{CV} = \boldsymbol{V\Sigma}, \quad \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}.$$

Also, we have $\mathrm{Tr}(\boldsymbol{V}^\top \boldsymbol{CV}) = \mathrm{Tr}(\widetilde{\boldsymbol{U}}^\top \boldsymbol{U}^\top \boldsymbol{CU}\widetilde{\boldsymbol{U}}) = \mathrm{Tr}(\boldsymbol{U}^\top \boldsymbol{CU})$. Thus, we can safely restrict the search space to the set of the solutions satisfying Eq. (2.9), we can see that Eq. (2.7) reduces to

$$\operatorname*{argmax}_{(\boldsymbol{V},\boldsymbol{\Sigma})\in \mathrm{EigPair}(\boldsymbol{C})} \mathrm{Tr}(\boldsymbol{\Sigma})$$

where $\mathrm{EigPair}(\boldsymbol{C}) := \{(\boldsymbol{V}, \boldsymbol{\Sigma}) \in \mathbb{R}^{d\times k} \times \mathbb{R}^{k\times k} \mid \boldsymbol{CV} = \boldsymbol{V\Sigma}, \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}, \boldsymbol{\Sigma} \text{ is a diagonal matrix}\}$ is the set of all matrices satisfying Eq. (2.9). This means that the maximum is attained when $\boldsymbol{V}$ is the concatenation of the eigenvectors corresponding to the k largest eigenvalues of $\boldsymbol{C}$, and $\boldsymbol{U} = \boldsymbol{V}\widetilde{\boldsymbol{U}}^\top$ maximizes Eq. (2.7), where $\widetilde{\boldsymbol{U}} \in \mathbb{R}^{d\times k}$ is an unidentified orthogonal

matrix, but the optimal projection $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{V}\boldsymbol{V}^\top$ does not depend on $\widetilde{\boldsymbol{U}}$ and can be calculated.

2.4.3 Density Estimation

Density estimation is the fundamental problem of estimating the probability density function p_x from i.i.d. data following it: $\{x_i\}_{i=1}^n \overset{\text{i.i.d.}}{\sim} p_x$. It is an unsupervised problem in that the actual density values are not given along with training data but have to be inferred from the unlabeled data.

Kernel Density Estimator (KDE) (Loftsgaarden and Quesenberry, 1965; Rosenblatt, 1956) is a well-known nonparametric density estimation method. It predicts the density at an arbitrary test point by averaging the weights given by normalized kernel functions each centered at a training point. More formally, the KDE is the following estimator:

$$\widehat{p}_x(x) = \frac{1}{n} \sum_{i \in [n]} k(x, x_i),$$

where $k : \mathbb{R}^d \to \mathbb{R}$ is a positive-definite kernel function that is positive and normalized: $\int k(x, x')\mathrm{d}x = 1$.

Since the probability density function tells a lot about the population, numerous problems can be reduced to this fundamental problem. However, due to its generality, density estimation tends to be hard compared to more task-specific problems especially in high-dimensional cases, and practitioners are advised to rather avoid solving this problem, if possible.

2.4.4 Clustering

Clustering is an unsupervised learning problem of grouping data points so that similar points will be assigned to the same group and dissimilar ones will not belong to different ones. It is often used to visualize and interpret data as well as for classification when no supervised label is available in a training data set.

2.4.4.1 K-Means Clustering

The *K-means clustering*[*7] is a popular clustering method used when the number of the clusters K is known in advance. It literally maintains the means of K clusters and assigns each data point to the closest mean.

More specifically, the K-means clustering tries to find the minimizer of the following optimization problem:

$$\underset{c:\mathbb{R}^d \to [K]}{\operatorname{argmin}} \underset{\boldsymbol{x} \sim D}{\mathbf{E}} [\|\boldsymbol{x} - \mu_{c(\boldsymbol{x})}\|^2], \tag{2.9}$$

[*7] Hans-Hermann (2008) mentions the origin of K-means as follows: "When tracing back this algorithm to its origins, we see that it has been proposed by several scientists in different forms and under different assumptions."

where $\mu_k = 1, \ldots, K$ be index of the clusters, c is a function that assigns every data point to one of the K clusters, and $\mu_k = \mathbf{E}[\boldsymbol{x} \mid c(\boldsymbol{x}) = k]$.

We are often interested in solving a *transductive* version of this problem where the expectation is taken over the empirical distribution on the training data:

$$\min_{c:\mathcal{S}_{\mathrm{tr}} \to [K]} \frac{1}{n} \sum_{i \in [n]} \|\boldsymbol{x}_i - \overline{\boldsymbol{x}}_{c(\boldsymbol{x}_i)}\|^2. \tag{2.10}$$

Here, c can be restricted on $\mathcal{S}_{\mathrm{tr}}$, and $\overline{\boldsymbol{x}}_k := \frac{1}{S_k} \sum_{\boldsymbol{x} \in S_k} \boldsymbol{x}$ with $S_k := \{\boldsymbol{x} \in \mathcal{S}_{\mathrm{tr}} : c(\boldsymbol{x}) = k\}$.

We can find many extensions of the method (Arthur and Vassilvitskii, 2007; Hans-Hermann, 2008).

2.4.5 Mode-Seeking Clustering

A drawback of K-means clustering is that we need to know the number of clusters as prior knowledge, which may not be the case in some applications. In contrast, methods based on the *mode-seeking clustering* approach do not require us to specify the number of clusters (Cheng, 1995; Comaniciu and Meer, 2002; Fukunaga and Hostetler, 1975; Sasaki et al., 2014). *Mode-seeking clustering* methods seek a mode close to each data point by iteratively moving the data point in the direction of the gradient of the data density.

In this approach, accurate estimation of the gradient direction is a critical task for the clustering result. Cheng (1995); Comaniciu and Meer (2002); Fukunaga and Hostetler (1975) estimate the density function by the KDE first, and then calculate the gradient of the estimate.

Sasaki et al. (2014) proposed a clustering method based on estimation of the *log-density gradient*, i.e., the gradient of the logarithm of the density, where the estimation is directly targeted at the gradient unlike the two-stage, KDE-based approach, leading to empirically promising performance. Estimation of the log-density gradient was also studied in Beran (1976); Cox (1985).

2.5 Semi-Supervised Learning

In a few words, *semi-supervised learning* is supervised learning with additional unlabeled data. For example, in a typical semi-supervised classification problem, we are given unlabeled data $\{\boldsymbol{x}_i\}_{i=1}^{n_{\mathrm{U}}}$ additionally to labeled data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ (Chapelle et al., 2006; Sakai et al., 2017, 2018).

Semi-supervised learning methods are often used to alleviate the situation where we only have a limited amount of labeled data that are often expensive to collect, by exploiting information from cheap unlabeled data.

2.6 Weakly-Supervised Learning

Weakly-supervised learning is an intermediate scenario between supervised learning and unsupervised learning. In weakly-supervised learning, only part of data points are labeled by the output variable, or no output label is available but another kind of labels are given in the training phase. There is no direct supervision about the output variable unlike supervised learning, but there is limited supervision unlike unsupervised learning. As we will see below, the indirect supervision can be given in variety of forms depending on the problem.

2.6.1 Positive-Unlabeled Learning

Positive-Unlabeled (PU) learning is a binary classification problem where we are given positive data $\{x_i^{\mathrm{P}}\}_{i=1}^{n_{\mathrm{P}}} \overset{\text{i.i.d.}}{\sim} D_{x|y=+1}$ and unlabeled data $\{x_i^{\mathrm{U}}\}_{i=1}^{n_{\mathrm{U}}} \overset{\text{i.i.d.}}{\sim} D_x$ but *none* of negative data following $D_{x|y=-1}$ is available, where D denotes the distribution of (x, y).[*8] This setting has been actively studied recently (Blanchard et al., 2010; du Plessis et al., 2014, 2015; Elkan and Noto, 2008; Kiryo et al., 2017; Niu et al., 2016).

2.6.2 Unlabeled-Unlabeled Learning

Another interesting binary classification problem is *Unlabeled-Unlabeled (UU) learning* (du Plessis et al., 2013; Lu et al., 2018). In this problem, we only have unlabeled data, but they are given as two data sets $\{\boldsymbol{x}_i^1\}_{i=1}^{n_1} \overset{\text{i.i.d.}}{\sim} D^1$ and $\{\boldsymbol{x}_i^2\}_{i=1}^{n_2} \overset{\text{i.i.d.}}{\sim} D^2$ whose respective distributions D^1 and D^2 differ only in their class prior distributions: $D_y^1 \neq D_y^2$, but $D_{\boldsymbol{x}|y}^1 = D_{\boldsymbol{x}|y}^2$.*8 du Plessis et al. (2013) showed that under the assumption that the test distribution D^{te} has the balanced class prior distribution, i.e., $\Pr_{D^{\mathrm{te}}}[y = +1] = \Pr_{D^{\mathrm{te}}}[y = -1]$, the classification boundary $\{\boldsymbol{x} \mid \Pr_{D^{\mathrm{te}}}[y = +1 \mid \boldsymbol{x}] = \Pr_{D^{\mathrm{te}}}[y = -1 \mid \boldsymbol{x}]\}$ can be learned although the label, i.e., the sign of $\Pr_{D^{\mathrm{te}}}[y = +1 \mid \boldsymbol{x}] - \Pr_{D^{\mathrm{te}}}[y = -1 \mid \boldsymbol{x}]\}$ cannot be identified. Lu et al. (2018) relaxed these limitations and showed that the classifier (the predictor for the boundary and the sign) can be estimated for any test distribution D^{te} if we know the training class priors D_y^1 and D_y^2.

2.6.3 Comparison of Supervised, Semi-Supervised, Unsupervised, and Weakly Supervised Learning

Table 2.1 shows a brief comparison of forms of training and test data in supervised, semi-supervised, unsupervised, and weakly supervised learning. In supervised learning, we have output samples y_i and y_i' both in training and test data sets. In semi-supervised learning, we have additional unlabeled data x_i^{U} in a training data set. In unsupervised learning, no output sample is given as training data although a test data set may or may not have output samples. In weakly supervised learning, there is no output sample available in a training data set, but we are given another type of *weak* labels l_i that has information about the output instead. For example, PU learning can be seen as a weakly supervised learning

[*8] Refer to Section 2.1 for our notational convention.

Table 2.1: Comparison of the settings of supervised, unsupervised, and weakly-supervised learning from the aspect of information provided as training data as well as test data.

	Training Data	Test Data
Supervised learning	$\{(x_i, y_i)\}_{i=1}^{n}$	$\{(x'_i, y'_i)\}_{i=1}^{n'}$
Semi-supervised learning	$\{(x_i, y_i)\}_{i=1}^{n}$ and $\{x_i\}_{i=n+1}^{m}$	$\{(x'_i, y'_i)\}_{i=1}^{n'}$
Unsupervised learning	$\{x_i\}_{i=1}^{n}$	$\{(x'_i, y'_i)\}_{i=1}^{n'}$ or $\{x'_i\}_{i=1}^{n'}$
Weakly Supervised learning	$\{(x_i, l_i)\}_{i=1}^{n}$	$\{(x'_i, y'_i)\}_{i=1}^{n'}$

with $l_i \in \{+1, 0\}$, where $l_i = +1$ indicates that instance x_i is from the positive class, and 0 indicates that x_i can be a positive or a negative instance. For UU learning, we can define $l_i \in \{1, 2\}$ indicating that instance x_i is drawn from the distribution D^{l_i}.

2.7 Transfer Learning

Transfer learning, in a broad sense, refers to the special case of learning from limited information where all or most, depending on the specific setting, of training data are qualitatively different from those used in the test environment for performance evaluation. Here, information is limited in the sense that there are few or none of data available for training that are directly generated from the test environment. In such cases, naively minimizing the empirical risk on the training data would result in poor performance in the test environment. A challenge is to devise a way to fill the gap by transferring knowledge from the training domain to the test domain so that the learner can perform well in the latter domain.

An extensive survey on transfer learning can be found in Pan and Yang (2010).

2.7.1 Domain Adaptation

Domain adaptation is learning under the presence of distribution shift between the training and the test domain although the types of the training and the test data are the same. When a model is pre-trained on data from the training domain and then slightly calibrated to the test domain, the calibration part is called *fine tuning* in the deep learning community, which can be seen as an instance of domain adaptation.

2.7.2 Covariate Shift Adaptation

Covariate shift is a domain adaptation setting where the covariate (or input) distribution changes in the training and the test phase: $p_x^{\mathrm{tr}} \neq p_x^{\mathrm{te}}$, where p^{tr} is the probability density function for the training, and p^{te} is that for the test. The goal is to estimate a function characterized by $p_{y|x}^{\mathrm{te}}(y \mid x)$ (e.g., regression function) from data following the training distribution, $\{(x_i, y_i)\}_{i=1}^{n} \sim p_{x,y}^{\mathrm{tr}}(x, y)$, under the assumption that $p_{y|x}^{\mathrm{tr}}(y \mid x) = p_{y|x}^{\mathrm{te}}(y \mid x)$ (Shimodaira, 2000; Sugiyama and Kawanabe, 2012).

It has been shown that in many cases, learning performance improves by weighting the loss by the ratio of the test density to the training density: $p_x^{\mathrm{te}}(x)/p_x^{\mathrm{tr}}(x)$ in empirical risk minimization. (Shimodaira, 2000; Sugiyama and Kawanabe, 2012; Sugiyama et al., 2012).

2.7.2.1 Output Distribution Shift Adaptation

This is another domain adaptation setting where the marginal distribution of the output variable differs for training and test: $p_{\mathrm{tr}}(y) \neq p_{\mathrm{te}}(y)$, but we assume that $p_{x|y}^{\mathrm{tr}} = p_{x|y}^{\mathrm{te}}$. The goal is to estimate a function characterized by $p_{\mathrm{te}}(y \mid x)$ (e.g., a classifier, a regression function) from data following the training distribution, $\{(x_i, y_i)\}_{i=1}^n \sim p_{\mathrm{tr}}(x, y)$ (du Plessis and Sugiyama, 2014; Kawakubo et al., 2016).

2.7.3 Multi-Task Learning

In *multi-task learning*, we are given multiple learning tasks to be solved. Those tasks have their respective datasets and learning targets that are (in general) different from each other. When the learning targets are related to each other, jointly solving the tasks while sharing information across the tasks may lead to better learning results than those when solved separately.

For this reason, empirical and theoretical studies on multi-task learning have been actively conducted (Ando and Zhang, 2005; Baxter, 2000; Caruana, 1998; Evgeniou and Pontil, 2004a; Thrun, 1996; Zhang, 2013).

Caruana (1998) proposed a neural network architecture for multi-task learning whose lower-layers are shared by multiple tasks while higher-level layers being task-specific and independent. This approach of explicitly sharing part of learning parameters among tasks is called *hard parameter-sharing*. It is one of the oldest multi-task method in the machine learning literature.

Evgeniou and Pontil (2004a) proposed a simple yet practical approach called *regularized multi-task learning*, which uses a regularizer that encourage the solutions of similar tasks to be close to each other. This approach of mildly imposing task relatedness by regularizers, without explicitly sharing learning parameters is called *soft parameter-sharing* as opposed to *hard parameter-sharing*. In this approach, the optimization problem usually looks as follows:

$$\min_{f_1,\ldots,f_T} \sum_{t=1}^{T} [\mathcal{L}_t(f_t) + \Omega_t(f_t)] + \Omega(f_1, \ldots, f_T),$$

where f_t is the model, $\mathcal{L}_t$ is the loss, and Ω_t is the regularizer for the t-th task. Ω is the regularizer for incorporating the task relationship. The hard parameter-sharing approach can be too restrictive in some cases where the tasks are similar but not to the extent where they share identical parameters. The *soft weight sharing approach* relaxes the hard weight sharing by penalizing the discrepancy between parameters across tasks instead of enforcing them to be exactly the same.

Thrun (1996) proposed the *lifelong learning framework*, where the learning task to be solved changes over time, and the knowledge obtained from the past tasks is transfered to subsequent tasks, whose applications include image recognition.

Baxter (2000) defined a theoretical framework called *inductive bias learning*, and gave a theoretical analysis for a generalization error bound for a class of multi-task learning methods.

The semi-supervised multi-task learning method proposed by Ando and Zhang (2005) generates many auxiliary learning tasks from unlabeled data and seeks a good feature mapping for the target learning task within a similar framework of inductive bias learning (Baxter, 2000).

2.8 Causal Inference and Uplift Modeling

This section introduces some basics on causal inference. Although there are several frameworks for formal treatment of causality, we only cover the *potential outcome* framework introduced in Rubin (2005). Many other interesting and advanced topics on causal inference can be found in Hernán and Robins (2018); Pearl (2009).

2.8.1 Potential Outcomes

In the potential outcome framework, we have two real-valued random variables $y(-1)$ and $y(1)$ that are not observed at the same time, but only one of them can be observed.[*9] We call these variables *potential outcomes*. When one of the two variables is observed, the observed one is called the *factual* outcome, and the other one is called the *counter-factual* outcome. For example, in a medical treatment application, the argument of $y(\cdot)$ may represent whether the medical treatment has been given to a patient (1 meaning that it has been, and -1 otherwise), and the observation of $y(\cdot)$ may represent how much the health condition is improved. The two actions of treating (1) and not treating (-1) cannot be taken at the same time, and only the potential outcome corresponding to the chosen action can ever be observed.

2.8.2 Treatment Variable

Let t be another variable called *treatment variable* taking its value on $\{-1, 1\}$. Suppose that t represents a stochastic decision about whether to treat ($t = 1$) or not ($t = -1$) and thus decides which outcome, $y(1)$ or $y(-1)$ respectively, to be observed. In other words, we observe $y(\underline{t})$ but never $y(-\underline{t})$ when the treatment assignment is $t = \underline{t}$ for $\underline{t} \in \{-1, 1\}$. Then, the factual outcome under the treatment assignment by t can be written as $y(t)$, and the counter-factual one as $y(-t)$.

[*9] We consider real-valued variables for simplicity.

2.8.3 Treatment Effect

Knowing the *treatment effect*, or *causal effect*, defined as $y(1) - y(-1)$, would be often of interest in real-world applications since it directly tells which action, treating (1) or not treating (-1), results in the better outcome as well as how better it is. However, without any further assumption, it is impossible to know this quantity due to the exclusive nature in observation of the potential outcomes. This issue is referred to as the *fundamental problem of causal inference* (Holland, 1986).

Instead of the treatment effect defined above, we may be interested in estimating the *average treatment effect* defined as $\mathbf{E}[y(1)] - \mathbf{E}[y(-1)]$, where the expectation is taken over all possibilities of $y(1)$ and $y(-1)$. Intuitively, it seems possible to estimate the expectations by collecting many samples that are given the treatment 1 and -1 respectively. This is true under some conditions, but not always. Below, we will consider when and how the estimation is possible.

2.8.4 Difference between Causation and Statistical Association

Causal notions and statistical notions are related to but different from each other. In particular, causal relationships and statistical association between variables can be confusing, and analyzing data without understanding their difference can critically mislead interpretations and conclusions.

To see this in a simple example, we show the following paradoxical mathematical fact about the average treatment effect that can be shown under the potential outcome framework.

Example 2.8.1. Consider the following medical treatment example: a patient receives the medical treatment determined by the variable t, and we observe the counter factual outcome $y(t)$ as a consequence. Suppose that there are only four patient in the world, and we have observed t and $y(t)$ for all of them. The variables take values for each patient as in Table 2.2. Note that neither $y(1)$ nor $y(-1)$ are directly observed although either could be indirectly observed through $y(t)$.

Now, we want to know the effect of the medical treatment. It might be tempting to say that it can be calculated the difference of the conditional expectation $\mathbf{E}[y(t) \mid t = 1] - \mathbf{E}[y(t) \mid t = -1]$, which is, however, incorrect. If we calculate it, it gives $\mathbf{E}[y(t) \mid t = 1] - \mathbf{E}[y(t) \mid t = -1] = 2/2 - 1/2 = 1/2$. On the other hand, the true average treatment effect is $\mathbf{E}[y(1)] - \mathbf{E}[y(-1)] = 2/4 - 2/4 = 0$, i.e., there is no treatment effect on average.

The reason why this counter-intuitive phenomenon occurs is because t has statistical dependency on $y(1)$. In the example above, t is correlated with $y(1)$: $t = 1$ if $y(1) = 1$ and -1 otherwise, which biases the conditional expectation $\mathbf{E}[y(t) \mid t = 1]$. From the table, we can also see that t is statistically independent of $y(1)$, and $\mathbf{E}[y(t) \mid y = -1]$ is equal to $\mathbf{E}[y(-1)]$. As we will see later, this is not a coincidence.

□

Table 2.2: The population for Example 2.8.1. t and $y(t)$ are observed. Numbers in parentheses are not directly observed.

	Patient 1	Patient 2	Patient 3	Patient 4	Average
$y(1)$	(0)	(1)	(0)	(1)	(2/4)
$y(-1)$	(0)	(0)	(1)	(1)	(2/4)
t	-1	1	-1	1	
$y(t)$	0	0	1	1	
$y(t) \mid t = 1$		1		1	2/2
$y(t) \mid t = -1$	0		1		1/2

Example 2.8.1 illustrates that the average treatment cannot be always estimated by conditional expectations. In other words, we cannot safely make causal inference using statistical techniques without any assumption. Are there any conditions for validating the average treatment effect estimation by based on the conditional expectations? In the next subsection, we introduce some sufficient conditions for this.

2.8.5 Condition for Statistical Average Treatment Effect Estimation

In Example 2.8.1, $\mathbf{E}[y(1) \mid t = 1]$ failed to estimate $\mathbf{E}[y(1)]$. This was because t and $y(1)$ had some statistical dependency. A natural question is, what if it was not the case?

Definition 2.8.1 (Exchangeability)**.** When $y(\underline{s}) \perp\!\!\!\perp t$ for all $\underline{s} \in \{-1, 1\}$, where $(\cdot) \perp\!\!\!\perp (\cdot)$ indicates the independence of variables, we say that they satisfy *exchangeability.*[*10]:

In other words, the exchangeability states that the treatment variable t is (statistically) dependent on *neither* of the two potential outcomes $y(1)$ nor $y(-1)$.

Proposition 2.8.1. *When the exchangeability holds, we have*

$$\begin{aligned}\mathbf{E}[y(t) \mid t = \underline{s}] &= \mathbf{E}[y(\underline{s}) \mid t = \underline{s}] \\ &= \mathbf{E}[y(\underline{s})],\end{aligned}$$

for $\underline{s} = -1, 1$.

2.8.6 Controlled Randomized Trials and Observational Studies

Experiments or trials with a completely randomized treatment variable (with no dependence on any other variables) under a controlled environment are called *Controlled Randomized Trials (CRTs).* CRTs are useful and often conducted since samples collected from a CRT

[*10] Pearl (2009) explains the nuance of the term 'exchangeability' as follows: "the investigator is instructed to imagine a hypothetical *exchange* of the two groups (the treated group becomes untreated, and vice versa) and then to judge whether the observed data under the swap would be distinguishable from the actual data."

satisfy the exchangeability, and thus the average treatment effect can be estimated as explained above.

While causal inference using CRT samples is a simple and solid approach, it is not always feasible. A reason is that the treatment subject to an investigation cannot always be controlled or randomized due to technical, ethical, or economical reasons. In a medical treatment example, some patients may be unwilling to take well-tested medical treatments just because they are randomly chosen.

For this reason, methodologies for causal inference using data collected from observations without controlling treatment assignments are practically important and have been actively explored (Gutierrez and Gérardy, 2017; Rosenbaum, 2010; Shalit et al., 2017). Investigations and analyses from such data are called *observational studies*.

2.8.7 Conditional Average Treatment Effect

When we have access to some covariate x that may potentially affect the causal relationship between y and t, finer analyses may be possible by knowing the conditional average treatment effect. We define the *conditional average treatment effect (CATE)* as follows:

$$u(\underline{x}) := \mathbf{E}[y(1) \mid x = \underline{x}] - \mathbf{E}[y(-1) \mid x = \underline{x}].$$

In this dissertation, we also use *individual treatment effect (ITE)* and *individual uplift* for referring to conditional average treatment effect.

In parallel to the case of the average treatment effect, we have the following sufficient condition for the identifiability of the conditional average treatment effect:

Definition 2.8.2 (Conditional Exchangeability)**.** Suppose that $y(\underline{t}) \perp\!\!\!\perp t \mid x = \underline{x}$ for all $\underline{t} \in \{-1, 1\}$ and all $x \in \mathcal{X}$, where $(\cdot) \perp\!\!\!\perp (\cdot) \mid (\cdot)$ indicates the independence of the first two variables conditioned on the third variable, and $\mathcal{X}$ is the range of x. Then, we say that t is *exchangeable with respect to y conditioned on x.*

Proposition 2.8.2 (Wasserman (2013))**.** *When t is exchangeable with respect to y conditioned on x, we have*

$$\begin{aligned}\mathbf{E}[y(t) \mid t = \underline{t}, x = \underline{x}] &= \mathbf{E}[y(\underline{t}) \mid t = \underline{t}, x = \underline{x}] \\ &= \mathbf{E}[y(\underline{t}) \mid x = \underline{x}],\end{aligned}$$

for all $\underline{t} \in \{-1, 1\}$ and all $\underline{x} \in \mathcal{X}$.

Corollary 2.8.1. *Under the assumptions of Proposition 2.8.2, we have*

$$u(\underline{x}) := \mathbf{E}[y(t) \mid t = 1, x = \underline{x}] - \mathbf{E}[y(t) \mid t = -1, x = \underline{x}].$$

2.8.8 Uplift Modeling

In many real-world problems, we are required to optimize an action or a treatment so as to maximize some profit. *Uplift modeling* is the field studying such decision making problems based on machine learning approaches (Gutierrez and Gérardy, 2017; Jaskowski and Jaroszewicz, 2012; Radcliffe and Surry, 1999, 2011; Rzepakowski and Jaroszewicz, 2012a; Shalit et al., 2017).

In uplift modeling, the individual uplift $u(x)$ plays a very important role, and thus uplift modeling is closely related to treatment effect estimation. We can see that once we estimate $u(x)$, it provides several useful hints about our decision making as follows.

We define a *treatment policy* $\pi(t \mid x)$ by a conditional probability density of t given x, which represents our stochastic decision making rule about the treatment assignment after observing x. The average of the outcome y when the treatment t follows the treatment policy $\pi(t \mid x)$ is

$$\begin{aligned} U(\pi) &:= \iint \sum_{t=-1,1} y p(y \mid x, t) \pi(t \mid x) p(x) \mathrm{d}y \mathrm{d}x \\ &= \underbrace{\int u(x) \pi(t = 1 \mid x) p(x) \mathrm{d}x}_{=:U(\pi)} + \underbrace{\iint \sum_{t=-1,1} y p(y \mid x, t) 1[t = -1] p(x) \mathrm{d}y \mathrm{d}x}_{\text{Constant w.r.t. } \pi} . \end{aligned}$$

If we want to maximize this quantity with respect to π, we only have to maximize the first term $U(\pi)$. Hence, an optimal solution is given by $\pi(t = 1 \mid x) = 1[0 \leq u(x)]$. Note that $\pi(t = 1 \mid x) \in [0, 1]$. This means that the optimal treatment policy maximizing the average outcome can be easily obtained if we know $u(x)$.

Furthermore, $u(\boldsymbol{x})$ can be used for ranking individuals to decide what individuals should be prioritized to be treated. This is especially useful when the treatment incurs some cost and we have a limited budgets for paying the cost. In this case, we need to select a limited number of individuals to be treated.

Suppose that we rank individuals according to some scoring function $f(x)$ and take the ones whose scores are more than or equal to some threshold $\alpha \in \mathbb{R}$. This corresponds to using the treatment policy $\pi_{f,\alpha}(t = 1 \mid x) := 1[\alpha \leq f(x)]$. Similarly we define $\pi_{u,\beta}(t = 1 \mid x) := 1[\beta \leq u(x)]$. If we compare the average outcome of $\pi_{f,\alpha}(t = 1 \mid x)$ and that of $\pi_{u,\beta}(t = 1 \mid x)$ under the condition that they treat the same portion of individual:

$\Pr[\alpha \leq f(x)] = \Pr[\beta \leq u(x)]$, their difference is

$$\begin{aligned}
U(\pi_{u,\beta}) - U(\pi_{f,\alpha}) &= \int u(x) \times 1[\beta \leq u(x)]p(x)\mathrm{d}x - \int u(x) \times 1[\alpha \leq f(x)]p(x)\mathrm{d}x \\
&\geq \int \beta \times 1[\beta \leq u(x)]p(x)\mathrm{d}x - \int u(x) \times 1[\alpha \leq f(x)]p(x)\mathrm{d}x \\
&= \beta \Pr[\beta \leq u(x)] - \int u(x) \times 1[\alpha \leq f(x)]p(x)\mathrm{d}x \\
&= \beta \Pr[\alpha \leq f(x)] - \int u(x) \times 1[\alpha \leq f(x)]p(x)\mathrm{d}x \\
&= \int \beta \times 1[\alpha \leq f(x)]p(x)\mathrm{d}x - \int u(x) \times 1[\alpha \leq f(x)]p(x)\mathrm{d}x \\
&= \int [\beta - u(x)] \times 1[\alpha \leq f(x)]p(x)\mathrm{d}x \\
&\geq 0,
\end{aligned}$$

where the lower bound 0 is attained when $\beta \leq u(x) \iff \alpha \leq f(x)$. This means that $\pi_{u,\beta}$ maximizes the average outcome under the constraint on the proportion of treated individuals. This holds for any threshold β. In this sense, $u(x)$ yields the optimal ranking scores.

This is related to the fact that $u(x)$ maximizes *Area Under the Uplift Curve (AUUC)*, which is a standard performance measure for uplift modeling methods (Jaskowski and Jaroszewicz, 2012; Radcliffe, 2007; Radcliffe and Surry, 2011; Rzepakowski and Jaroszewicz, 2012a). We discuss this topic in detail in Chapter 5.

Related Areas Uplift modeling has a close connection to causal effect/treatment effect estimation that has been studied in the causal inference literature, and the mathematical formulation of problems and the methodologies in the both domains have considerable overlaps. The focus of uplift modeling is more on the prediction about the outcome of the unseen or future situations and the optimization of our actions. On the other hand, the causal inference literature often focuses more on analysis, reasoning, or discovery of past events in the aspect of the causation mechanisms behind them.

It is also related to *off-line batch* multi-armed bandit problems (Li et al., 2010, 2011). In the standard multi-armed bandit problem, the agent or the player is allowed to actively interacts with the environment in an online manner. The agent chooses an action from multiple candidates and receives a reward that depends on the action from the environment. The goal is to take good actions in the long run, i.e., maximizing the cumulative reward. The agent must care about the efficiency in the whole course of the learning process including the learning process since actions of the agent are evaluated and rewarded even during the learning phase. In off-line batch multi-armed bandit problems, on the other hand, the agent does not interact with the environment in an online manner. Training is performed using data observed in games played by other agents with policies different from the policy being trained. In this case, we only care about the quality of the learning result. The overall

objective is common with uplift modeling: It is to obtain a good policy using data in an off-line manner.

Chapter 3

Regularized Multi-Task Learning for Multi-Dimensional Log-Density Gradient Estimation

Log-density gradient estimation is a fundamental statistical problem and possesses various practical applications such as clustering and measuring non-Gaussianity. A naive two-step approach of first estimating the density and then taking its log-gradient is unreliable because an accurate density estimate does not necessarily lead to an accurate log-density gradient estimate. To cope with this problem, a method to *directly* estimate the log-density gradient without density estimation has been explored, and demonstrated to work much better than the two-step method. The objective of this work is to further improve the performance of this direct method in multi-dimensional cases. Our idea is to regard the problem of log-density gradient estimation in each dimension as a task, and apply *regularized multi-task learning* to the direct log-density gradient estimator. We experimentally demonstrate the usefulness of the proposed multi-task method in log-density gradient estimation and mode-seeking clustering.

3.1 Introduction

Multi-task learning is a paradigm of machine learning for solving multiple related learning tasks simultaneously with the expectation that information brought by other related tasks can be mutually exploited to improve the accuracy (Caruana, 1997). Multi-task learning is particularly useful when one has many related learning tasks to solve but only few training samples are available for each task, which is often the case in many real-world problems such as therapy screening (Bickel et al., 2008) and face verification (Wang et al., 2009).

Multi-task learning has been gathering a great deal of attention, and extensive studies have been conducted both theoretically and experimentally (Ando and Zhang, 2005; Baxter, 2000; Evgeniou and Pontil, 2004a; Thrun, 1996; Zhang, 2013). Thrun (1996) proposed the *lifelong learning framework*, which transfers the knowledge obtained from the tasks experienced in the past to a newly given task, and it was demonstrated to improve the

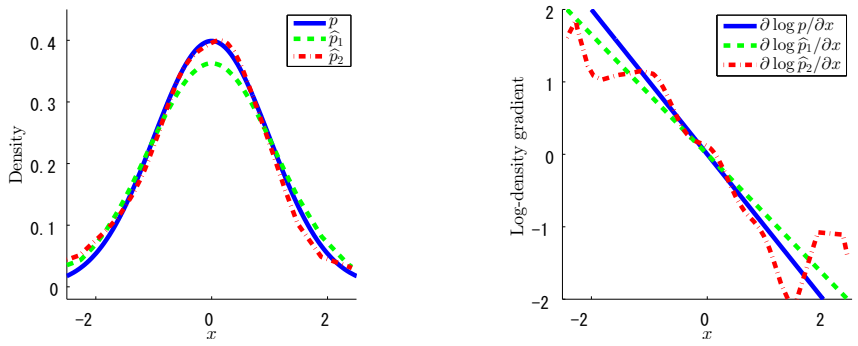
performance of image recognition. Baxter (2000) defined a multi-task learning framework called *inductive bias learning*, and derived a generalization error bound. The semi-supervised multi-task learning method proposed by Ando and Zhang (2005) generates many auxiliary learning tasks from unlabeled data and seeks a good feature mapping for the target learning task. Among various methods of multi-task learning, one of the simplest and most practical approaches would be *regularized multi-task learning* (Evgeniou et al., 2005; Evgeniou and Pontil, 2004a), which uses a regularizer that imposes the solutions of related tasks to be close to each other. Thanks to its generic and simple formulation, regularized multi-task learning has been applied to various types of learning problems such as regression and classification (Evgeniou et al., 2005; Evgeniou and Pontil, 2004a). In this chapter, we explore a novel application of regularized multi-task learning to the problem of *log-density gradient estimation* (Beran, 1976; Cox, 1985; Sasaki et al., 2014).

The goal of log-density gradient estimation is to estimate the gradient of the logarithm of an unknown probability density function using samples following it. Log-density gradient estimation has various applications such as clustering (Cheng, 1995; Comaniciu and Meer, 2002; Fukunaga and Hostetler, 1975; Sasaki et al., 2014), measuring non-Gaussianity (Huber, 1985) and other fundamental statistical topics (Singh, 1977).

Beran (1976) proposed a method for *directly* estimating gradients without going through density estimation, to which we refer as *least-squares log-density gradients* (LSLDG). This direct method was experimentally shown to outperform the naive one consisting of density estimation followed by log-gradient computation, and was demonstrated to be useful in clustering (Sasaki et al., 2014).

The objective of this work is to estimate log-density gradients further accurately in multi-dimensional cases, which is still a challenging topic even using LSLDG. It is important to note that since the output dimensionality of the log-density gradient $\nabla \log p(\boldsymbol{x})$ is the same as its input dimensionality d, multi-dimensional log-density gradient estimation can be regarded as having multiple learning tasks if we regard estimation of each output dimension as a task. Based on this view, in this work, we propose to apply regularized multi-task learning to LSLDG. We also provide a practically useful design of parametric models for successfully applying regularized multi-task learning to log-density gradient estimation. We experimentally demonstrate that the accuracy of LSLDG can be significantly improved by the proposed multi-task method in multi-dimensional log-density estimation problems and that a mode-seeking clustering method based on the proposed method outperforms other methods.

The organization of this chapter is as follows: In Section 3.2, we formulate the problem of log-density gradient estimation and review LSLDG. Section 3.3 reviews the core idea of regularized multi-task learning. Section 3.4 presents our proposed log-density gradient estimator and algorithms for computing the solution. In Section 3.5, we experimentally demonstrate that the proposed method performs well on both artificial and benchmark data. Application to mode-seeking clustering is given in Section 3.6. Section 3.7 concludes this chapter with potential extensions of our work.



(A) Density estimation (B) Log-density gradient estimation

Figure 3.1: A comparison of two log-density gradient estimates based on density estimation. In (a), $\widehat{p}_2$ is a better estimate to the true density p than $\widehat{p}_1$, while in (b), $\nabla \log \widehat{p}_1$ is a better estimate to the true log-density gradient $\nabla \log p$ than $\nabla \log \widehat{p}_2$.

3.2 Log-density gradient estimation

In this section, we formulate the problem of *log-density gradient estimation*, and then review LSLDG.

3.2.1 Problem formulation and a naive method

Suppose that we are given a set of samples, $\{\boldsymbol{x}_i\}_{i=1}^n$, which are independent and identically distributed from a probability distribution with unknown density $p(\boldsymbol{x})$ on $\mathbb{R}^d$. The problem is to estimate the gradient of the logarithm of the density $p(\boldsymbol{x})$ from $\{\boldsymbol{x}_i\}_{i=1}^n$:

$$\nabla \log p(\boldsymbol{x}) = (\partial_1 \log p(\boldsymbol{x}), \ldots, \partial_d \log p(\boldsymbol{x}))^\top = \left(\frac{\partial_1 p(\boldsymbol{x})}{p(\boldsymbol{x})}, \ldots, \frac{\partial_d p(\boldsymbol{x})}{p(\boldsymbol{x})} \right)^\top,$$

where ∂_j denotes the partial derivative operator $\partial/\partial x^{(j)}$ for $\boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})^\top$.

A naive method for estimating the log-density gradient is to first estimate the probability density, which is performed by, e.g., kernel density estimation (KDE) as

$$\widehat{p}(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{2\sigma^2} \right),$$

where $\sigma > 0$ denotes the Gaussian bandwidth, then to take the gradient of the logarithm of $\widehat{p}(\boldsymbol{x})$ as

$$\partial_j \log \widehat{p}(\boldsymbol{x}) = \frac{\partial_j \widehat{p}(\boldsymbol{x})}{\widehat{p}(\boldsymbol{x})}.$$

However, this two-step method does not work well because an accurate density estimate does not necessarily provide an accurate log-density gradient estimate. For example, Figure 3.1 illustrates that a worse (or better) density estimate can produce a better (or worse) gradient estimate.

To overcome this problem, LSLDG, a single-step method which directly estimates the gradient without going through density estimation, was proposed (Beran, 1976; Cox, 1985; Sasaki et al., 2014), and has been demonstrated to experimentally work well. Next, we review LSLDG.

3.2.2 Direct estimation of log-density gradients

The basic idea of LSLDG is to directly fit a model $g_j(\boldsymbol{x})$ to the true log-density gradient $\partial_j \log p(\boldsymbol{x})$ under the squared loss:

$$\begin{aligned}
R_j(g_j) &:= \int \left(g_j(\boldsymbol{x}) - \partial_j \log p(\boldsymbol{x})\right)^2 p(\boldsymbol{x}) \mathrm{d}x \\
&= \int \left(g_j(\boldsymbol{x}) - \frac{\partial_j p(\boldsymbol{x})}{p(\boldsymbol{x})}\right)^2 p(\boldsymbol{x}) \mathrm{d}x \\
&= \int g_j(\boldsymbol{x})^2 p(\boldsymbol{x}) \mathrm{d}x - 2 \int g_j(\boldsymbol{x}) \partial_j p(\boldsymbol{x}) \mathrm{d}x + C_j \\
&= \int g_j(\boldsymbol{x})^2 p(\boldsymbol{x}) \mathrm{d}x - 2 \int \left[g_j(\boldsymbol{x}) p(\boldsymbol{x})\right]_{x^{(j)}=-\infty}^{x^{(j)}=\infty} \mathrm{d}x^{(\backslash j)} + 2 \int \partial_j g_j(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}x + C_j \\
&= \int g_j(\boldsymbol{x})^2 p(\boldsymbol{x}) \mathrm{d}x + 2 \int \partial_j g_j(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}x + C_j,
\end{aligned}$$

where $C_j := \int \frac{(\partial_j p(\boldsymbol{x}))^2}{p(\boldsymbol{x})} \mathrm{d}x$ is a constant that does not depend on g_j, $\int (\cdot) \mathrm{d}x^{(\backslash j)}$ denotes integration except for $x^{(j)}$, and the last deformation comes from *integration by parts* under the mild condition that $g_j(\boldsymbol{x}) p(\boldsymbol{x}) \to 0$ as $|x^{(j)}| \to \infty$.

Then, the *LSLDG score* $J_j(g_j)$ is given as an empirical approximation to the risk $R_j(g_j)$ subtracted by C_j:

$$J_j(g_j) := \frac{1}{n} \sum_{i=1}^{n} g_j(\boldsymbol{x}_i)^2 + \frac{2}{n} \sum_{i=1}^{n} \partial_j g_j(\boldsymbol{x}_i). \tag{3.1}$$

As $g_j(\boldsymbol{x})$, a linear-in-parameter model is used:

$$g_j(\boldsymbol{x}) = \boldsymbol{\alpha}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x}) = \sum_{k=1}^{b} \alpha_j^{(k)} \psi_j^{(k)}(\boldsymbol{x}), \tag{3.2}$$

where $\alpha_j^{(k)}$ is a parameter, $\psi_j^{(k)}(\boldsymbol{x})$ is a differentiable basis function, and b is the number of the basis functions. By substituting (3.2) into (3.1) and adding an ℓ_2-regularizer, we can analytically obtain the optimal solution $\widehat{\boldsymbol{\alpha}}_j$ as

$$\begin{aligned}
\widehat{\boldsymbol{\alpha}}_j &= \arg\min_{\boldsymbol{\alpha}_j} \left[\boldsymbol{\alpha}_j^\top \boldsymbol{G}_j \boldsymbol{\alpha}_j + 2 \boldsymbol{h}_j^\top \boldsymbol{\alpha}_j + \lambda_j \|\boldsymbol{\alpha}_j\|^2\right] \\
&= -(\boldsymbol{G}_j + \lambda_j \boldsymbol{I}_b)^{-1} \boldsymbol{h}_j,
\end{aligned}$$

where $\lambda_j \geq 0$ is the regularization parameter, $\boldsymbol{I}_b$ is the $b \times b$ identity matrix, and

$$\boldsymbol{G}_j := \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\psi}_j(\boldsymbol{x}_i)\boldsymbol{\psi}_j(\boldsymbol{x}_i)^\top, \ \boldsymbol{h}_j := \frac{1}{n}\sum_{i=1}^{n} \partial_j \boldsymbol{\psi}_j(\boldsymbol{x}_i).$$

Finally, an estimator of the log-density gradient is obtained by

$$\hat{g}_j(\boldsymbol{x}) := \hat{\boldsymbol{\alpha}}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x}).$$

It was experimentally shown that LSLDG produces much more accurate estimates of log-density gradients than the KDE-based gradient estimator and that the clustering method based on LSLDG performs well (Sasaki et al., 2014).

3.3 Regularized multi-task learning

In this section, we review a multi-task learning framework called *regularized multi-task learning* (Evgeniou et al., 2005; Evgeniou and Pontil, 2004a), which is powerful and widely applicable to many machine learning methods.

Consider that we have T tasks of supervised learning as follows. The task t is to learn an unknown function $f_t^*(\boldsymbol{x})$ from samples of input-output pairs $\{(\boldsymbol{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^{n_t}$, where $y_i^{(t)}$ is the output $f_t^*(\boldsymbol{x})$ with noise at the input $\boldsymbol{x} = \boldsymbol{x}_i^{(t)}$. When $f_t^*(\boldsymbol{x})$ is modeled by a parameterized function $f_t(\boldsymbol{x}; \boldsymbol{\alpha}_t)$, learning is performed by finding the parameter $\boldsymbol{\alpha}_t$ which minimizes the empirical risk associated with some loss function $l(y, y')$:

$$\widehat{\boldsymbol{\alpha}}_t = \arg\min_{\boldsymbol{\alpha}_t} \frac{1}{n_t}\sum_{i=1}^{n_t} l(y_i^{(t)}, f_t(\boldsymbol{x}_i^{(t)}; \boldsymbol{\alpha}_t)) = \arg\min_{\boldsymbol{\alpha}_t} J_t(\boldsymbol{\alpha}_t),$$

where $J_t(\boldsymbol{\alpha}_t) = \sum_{i=1}^{n_t} l(y_i^{(t)}, f_t(\boldsymbol{x}_i^{(t)}; \boldsymbol{\alpha}_t))$.

In regularized multi-task learning, the objective function has regularization terms which impose every pair of parameters to be close to each other while $J_t(\boldsymbol{\alpha}_t)$ are jointly minimized:

$$\sum_{t=1}^{T} J_t(\boldsymbol{\alpha}_t) + \frac{1}{2}\gamma \sum_{t=1,t'=1}^{T} \gamma_{t,t'} \|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t'}\|^2,$$

where $\gamma \geq 0$ is the regularization parameter and $\gamma_{t,t'} \geq 0$ are the similarity parameters between the tasks t and t'.

It was experimentally demonstrated that the *multi-task support vector regression* (Evgeniou et al., 2005; Evgeniou and Pontil, 2004a), performs better than the single-task counterpart (Vapnik et al., 1997) especially when the tasks are highly related each other.

3.4 Proposed method

In this section, we present our proposed method and algorithms.

3.4.1 Basic idea

Our goal in this chapter is to improve the performance of LSLDG in *multi-dimensional* cases. For multi-dimensional input $\boldsymbol{x}$, the log-density gradient $\nabla \log p(\boldsymbol{x})$ has multiple output dimensions, meaning that its estimation actually consists of multiple learning tasks. Our basic idea is to apply regularized multi-task learning to solve these tasks simultaneously instead of learning them independently.

This idea is supported by the fact that the target functions of these tasks, $\partial_1 \log p(\boldsymbol{x})$, $\ldots$, $\partial_d \log p(\boldsymbol{x})$, are all derived from the same log-density $\log p(\boldsymbol{x})$, and thus they must be strongly related to each other. Under such strong relatedness, jointly learning them with sharing information with each other would improve estimation accuracy as has been observed in other existing multi-task learning work.

3.4.2 Regularized multi-task learning for least-squares log-density gradients (MT-LSLDG)

Here, we propose a method called *regularized multi-task learning for least-squares log-density gradients* (MT-LSLDG).

Our method MT-LSLDG is given by applying regularized multi-task learning to LSLDG. More specifically, we consider the problem of minimizing the following objective function:

$$\begin{aligned} J(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d) &= \sum_{j=1}^{d} J_j(g_j(\cdot; \boldsymbol{\alpha}_j)) + \sum_{j=1}^{d} \lambda_j \|\boldsymbol{\alpha}_j\|^2 + \frac{1}{2}\gamma \sum_{j,j'=1}^{d} \gamma_{j,j'} \|\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_{j'}\|^2 \\ &= \sum_{j=1}^{d} \left(\boldsymbol{\alpha}_j^\top \boldsymbol{G}_j \boldsymbol{\alpha}_j + 2\boldsymbol{\alpha}_j^\top \boldsymbol{h}_j + \lambda_j \|\boldsymbol{\alpha}_j\|^2 \right) + \frac{1}{2}\gamma \sum_{j,j'=1}^{d} \gamma_{j,j'} \|\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_{j'}\|^2, \quad (3.3) \end{aligned}$$

where the last term is the multi-task regularizer which imposes the parameters to be close to each other.

Denoting the minimizers of (3.3) by $\widehat{\boldsymbol{\alpha}}_1, \ldots, \widehat{\boldsymbol{\alpha}}_d$, the estimator $\widehat{\boldsymbol{g}}(\boldsymbol{x}) = (\widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_d(\boldsymbol{x}))^\top$ is given by, for $j = 1, \ldots, d$,

$$\widehat{g}_j(\boldsymbol{x}) = g_j(\boldsymbol{x}; \widehat{\boldsymbol{\alpha}}_j) = \widehat{\boldsymbol{\alpha}}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x}). \quad (3.4)$$

We call this method *regularized multi-task learning for least-squares log-density gradients* (MT-LSLDG).

3.4.3 The design of the basis functions

The design of the basis functions $\boldsymbol{\psi}_j(\boldsymbol{x})$ in MT-LSLDG is crucial to enjoy the advantage of regularized multi-task learning. A simple design would be to use a common function $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi^{(1)}(\boldsymbol{x}), \ldots, \phi^{(b)}(\boldsymbol{x}))$ for all $\boldsymbol{\psi}_j(\boldsymbol{x})$, that is, $\boldsymbol{\psi}_1(\boldsymbol{x}) = \cdots = \boldsymbol{\psi}_d(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})$. From (3.3) and (3.4), in this design, the multi-task regularizer promotes $g_j(\boldsymbol{x}; \widehat{\boldsymbol{\alpha}}_j)$ to be more close

to each other so that

$$g_1(\boldsymbol{x};\widehat{\boldsymbol{\alpha}}_1) \approx \cdots \approx g_d(\boldsymbol{x};\widehat{\boldsymbol{\alpha}}_d).$$

However, it is inappropriate that all $g_j(\boldsymbol{x};\widehat{\boldsymbol{\alpha}}_j)$ are similar because the different true partial derivatives, say $\partial_j \log p(\boldsymbol{x})$ and $\partial_{j'} \log p(\boldsymbol{x})$ for $j \neq j'$, show different profiles in general.

To avoid this problem, we propose to use the partial derivatives of $\boldsymbol{\phi}(\boldsymbol{x})$ as basis functions:

$$\boldsymbol{\psi}_j(\boldsymbol{x}) = \partial_j \boldsymbol{\phi}(\boldsymbol{x}). \tag{3.5}$$

Assuming that $\log p(\boldsymbol{x})$ is sufficiently smooth, a necessary condition of $g_j(\boldsymbol{x})$ approximating $\partial_j \log p(\boldsymbol{x})$ for all j and all $\boldsymbol{x}$ is that

$$\begin{aligned}\partial_{j'}\widehat{\boldsymbol{g}}_j(\boldsymbol{x}) &\approx \partial_{j'}\partial_j \log p(\boldsymbol{x}) \\ &= \partial_{j'}\partial_j \log p(\boldsymbol{x}) \approx \partial_j \widehat{\boldsymbol{g}}_{j'}(\boldsymbol{x}),\end{aligned}$$

i.e., $\partial_{j'}\widehat{\boldsymbol{g}}_j(\boldsymbol{x}) - \partial_j \widehat{\boldsymbol{g}}_{j'}(\boldsymbol{x}) \approx 0$. For the basis functions given by Eq. (3.5), this implies that

$$\begin{aligned}&\partial_{j'}\widehat{\boldsymbol{\alpha}}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x}) - \partial_j \widehat{\boldsymbol{\alpha}}_{j'}^\top \boldsymbol{\psi}_{j'}(\boldsymbol{x}) \approx 0, \\ &\text{i.e. } (\widehat{\boldsymbol{\alpha}}_j - \widehat{\boldsymbol{\alpha}}_{j'})^\top \partial_{j'}\partial_j \boldsymbol{\phi}(\boldsymbol{x}) \approx 0.\end{aligned} \tag{3.6}$$

The necessary condition Eq. (3.6) can be ensured by forcing the condition $\widehat{\boldsymbol{\alpha}}_j - \widehat{\boldsymbol{\alpha}}_{j'} \approx \mathbf{0}$ as long as $\|\partial_{j'}\partial_j \boldsymbol{\phi}(\boldsymbol{x})\|$ is bounded. Thus, it would be reasonable to encourage $\widehat{\boldsymbol{\alpha}}_j - \widehat{\boldsymbol{\alpha}}_{j'}$ to be close to zero by minimizing the multi-task regularizer in addition to the LSLDG objective function. Moreover, if

$$\int \partial_j \partial_{j'} \boldsymbol{\phi}(\boldsymbol{x}) \partial_j \partial_{j'} \boldsymbol{\phi}(\boldsymbol{x})^\top \mathrm{d}\boldsymbol{x} \tag{3.7}$$

is a positive semi-definite matrix, $\widehat{\boldsymbol{\alpha}}_j - \widehat{\boldsymbol{\alpha}}_{j'} \approx \mathbf{0}$ itself is a necessary condition for every $g_j(\boldsymbol{x})$ to be close to the respective target $\partial_j \log p(\boldsymbol{x})$. This means that it is always safe and desirable for the parameters of every task pair to be close to each other.

As a specific choice of $\phi^{(k)}(\boldsymbol{x})$, we use a Gaussian basis functions:

$$\begin{aligned}\psi_j^{(k)}(\boldsymbol{x}) &= \partial_j \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_k\|^2}{2\sigma^2}\right) \\ &= \frac{c_k^{(j)} - x^{(j)}}{\sigma^2} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_k\|^2}{2\sigma^2}\right),\end{aligned}$$

where $\boldsymbol{c}_k$ are the centers of the kernels, and $\sigma > 0$ is the Gaussian bandwidth parameter. In the case of the Gaussian basis functions, we can show that the corresponding matrix Eq will be positive semi-definite.

Algorithm 1 Similarity parameter tuning.

$\gamma_{j,j'} \leftarrow 1$ for every $j = 1, \ldots, d; j' = 1, \ldots, d$.
repeat
With the current values of $\gamma_{j,j'}$, calculate the estimates as

$$(\widehat{\boldsymbol{\alpha}}_1, \ldots, \widehat{\boldsymbol{\alpha}}_d) \leftarrow \arg \min_{(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d)} J(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d).$$

for $j = 1, \ldots, d; j' = 1, \ldots, d$ **do**

$$\gamma_{j,j'} \leftarrow \exp(-\|\widehat{\boldsymbol{\alpha}}_j - \widehat{\boldsymbol{\alpha}}_{j'}\|^2/\alpha^2). \tag{3.8}$$

end for
until $\gamma_{j,j'}$ converges for every $j, = 1, \ldots, d; j' = 1, \ldots, d$.

3.4.4 Hyper-parameter tuning

As in LSLDG, the hyper-parameters, which are the ℓ_2-regularization parameters λ_j, the Gaussian bandwidth σ, and the multi-task parameters $\gamma, \gamma_{j,j'}$, can be cross-validated in MT-LSLDG. The procedure of the K-fold cross-validation is as follows: First, we randomly partition the set of training samples S_{tr} into K folds $F_1, \ldots, F_K$. Next, for each $k = 1, \ldots, K$, we estimate the log-density gradient using the samples in $S_{\mathrm{tr}} \setminus F_k$, which is denoted by $\hat{g}_j^{(k)}$, and then calculate the LSLDG scores for the samples in F_k as $J_{\mathrm{CV}}^{(k)}$:

$$J_{\mathrm{CV}}^{(k)} = \frac{1}{|F_k|} \sum_{\boldsymbol{x} \in F_k} \widehat{g}_j^{(k)}(\boldsymbol{x})^2 + \frac{2}{|F_k|} \sum_{\boldsymbol{x} \in F_k} \frac{\partial \widehat{g}_j^{(k)}(\boldsymbol{x})}{\partial x^{(j)}}.$$

We average these LSLDG scores to obtain the K-fold cross-validated LSLDG score:

$$J_{\mathrm{CV}} = \frac{1}{K} \sum_{k=1}^{K} J_{\mathrm{CV}}^{(k)}.$$

Finally, we choose the hyper-parameters that minimize J_{CV}. Throughout this chapter, we set $K = 5$.

When the number of similarity parameters $\gamma_{j,j'}$ is large (i.e. the data dimensionality is high), cross-validation may be computationally inefficient. In this case, we may use the heuristic procedure which alternatingly updates the estimates $\widehat{\boldsymbol{\alpha}}_j$ and the similarity parameters $\gamma_{j,j'}$ as described in Algorithm 1, where α is a hyper-parameter to be selected by cross-validation. In the update formula (3.8), the procedure determines the similarity parameter $\gamma_{j,j'}$ to be used in the next iteration depending on how close the estimated parameters $\widehat{\boldsymbol{\alpha}}_j$ and $\widehat{\boldsymbol{\alpha}}_{j'}$ are.

3.4.5 Optimization algorithms in MT-LSLDG

Here, we develop two algorithms for minimizing (3.3). One algorithm is to directly evaluate the analytic solution and the other is an iterative method based on block coordinate

descent (Warga, 1963).

3.4.5.1 Analytic solution

For simplicity, we assume the similarity parameters are symmetric: $\gamma_{j,j'} = \gamma_{j',j}$. Then, the objective function $J(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d)$ can be expressed as a quadratic function in terms of $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1^\top, \ldots, \boldsymbol{\alpha}_d^\top)^\top$ as

$$J(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top (\boldsymbol{G} + \boldsymbol{C} \otimes \boldsymbol{I}_b)\boldsymbol{\alpha} + 2\boldsymbol{\alpha}^\top \boldsymbol{h},$$

where

$$\boldsymbol{G} = \mathrm{diag}(\boldsymbol{G}_1, \ldots, \boldsymbol{G}_d),\ \boldsymbol{h} = (\boldsymbol{h}_1^\top, \ldots, \boldsymbol{h}_d^\top)^\top,$$
$$\boldsymbol{C} := \mathrm{diag}(\lambda_1, \ldots, \lambda_d) + \gamma\, \mathrm{diag}\left(\sum_{j=1}^{d} \gamma_{1,j}, \ldots, \sum_{j=1}^{d} \gamma_{d,j}\right) - \gamma\boldsymbol{\Gamma},$$

$[\boldsymbol{\Gamma}]_{j,j'} = \gamma_{j,j'}$, $\mathrm{diag}(\cdot, \ldots, \cdot)$ is the block-diagonal matrix whose diagonal blocks are its arguments, and $\otimes$ denotes the Kronecker product. The minimizer $\widehat{\boldsymbol{\alpha}}$ of $J(\boldsymbol{\alpha})$ is analytically computed by

$$\widehat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) = -(\boldsymbol{G} + \boldsymbol{C} \otimes \boldsymbol{I}_b)^{-1}\boldsymbol{h}. \tag{3.9}$$

3.4.5.2 Block coordinate descent (BCD) method

Direct computation of the analytic solution (3.9) involves inversion of a $db \times db$ matrix. This may be not only expensive in terms of computation time but also infeasible in terms of memory space when the dimensionality d is very large.

Alternatively, we propose an algorithm based on block coordinate descent (BCD) (Warga, 1963). It is an iterative algorithm which only needs manipulation of a relatively small $b \times b$ matrix at each iteration. This alleviates the memory size requirement and hopefully reduces computation time if the number of iterations is not large.

A pseudo code of the algorithm is shown in Algorithm 2. At each update (3.10) in the algorithm, only one vector $\widetilde{\boldsymbol{\alpha}}_j$ is optimized in a closed-form while fixing the other parameters $\widetilde{\boldsymbol{\alpha}}_{j'}$ $(j' \neq j)$. The update (3.10) only requires computing the inverse of a $b \times b$ matrix, which seems to be computationally advantageous over evaluating the analytic solution in terms of the computation cost and memory size requirement.

Another important technique to reduce the overall computation time is to use *warm start* initialization: when the optimal value of γ is searched for by cross-validation, we may use the solutions $\widetilde{\boldsymbol{\alpha}}_1, \ldots, \widetilde{\boldsymbol{\alpha}}_d$ obtained with γ as initial values for another γ.

Algorithm 2 Block coordinate descent (BCD) algorithm.

Initialize $\widetilde{\boldsymbol{\alpha}}_1, \ldots, \widetilde{\boldsymbol{\alpha}}_d$.
repeat
  for $j = 1, \ldots, d$ **do**

$$\widetilde{\boldsymbol{\alpha}}_j \leftarrow \arg\min_{\boldsymbol{\alpha}_j} J(\widetilde{\boldsymbol{\alpha}}_1, \ldots, \widetilde{\boldsymbol{\alpha}}_{j-1}, \boldsymbol{\alpha}_j, \widetilde{\boldsymbol{\alpha}}_{j+1}, \ldots, \widetilde{\boldsymbol{\alpha}}_d)$$
$$= \left(\boldsymbol{G}_j + \lambda_j \boldsymbol{I}_b + 2\gamma \sum_{j \neq j'} \gamma_{j,j'} \boldsymbol{I}_b \right)^{-1} \left(-\boldsymbol{h}_j + 2\gamma \sum_{j' \neq j} \gamma_{j,j'} \widetilde{\boldsymbol{\alpha}}_{j'} \right). \tag{3.10}$$

  end for
until $\widetilde{\boldsymbol{\alpha}}_1, \ldots, \widetilde{\boldsymbol{\alpha}}_d$ converge.

3.5 Experiments on log-density gradient estimation

In this section, we illustrate the behavior of the proposed method and experimentally investigate its performance.

3.5.1 Experimental setting

In each experiment, training samples $\{\boldsymbol{x}_i\}_{i=1}^n$ and test samples $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ are drawn independently from an unknown density $p(\boldsymbol{x})$. We estimate $\nabla \log p(\boldsymbol{x})$ from the training samples, and then evaluate the estimation performance by the *test score*

$$J_{\mathrm{te}}(\widehat{\boldsymbol{g}}) = \sum_{j=1}^{d} \left[\frac{1}{n'} \sum_{i'=1}^{n'} \widehat{g}_j(\boldsymbol{x}'_{i'})^2 + \frac{2}{n'} \sum_{i'=1}^{n'} \partial_j \widehat{g}_j(\boldsymbol{x}'_{i'}) \right],$$

where $\widehat{\boldsymbol{g}}(\boldsymbol{x}) = (\widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_d(\boldsymbol{x}))^\top$ is an estimated log-density gradient. This score is an empirical approximation of the expected squared loss of $\widehat{\boldsymbol{g}}(\boldsymbol{x})$ over the test samples without the constant C_j (see Section 3.2.2), and a smaller score means a better estimate.

We compare the following three methods:

- The multi-task LSLDG (MT-LSLDG): our method proposed in Section 3.4.
- The single-task LSLDG (S-LSLDG): the existing method (Beran, 1976; Cox, 1985) reviewed in Section 3.2.2. This method agrees with MT-LSLDG at $\gamma = 0$.
- The common-parameter LSLDG (C-LSLDG): LSLDG with common parameters $\boldsymbol{\alpha}' = \boldsymbol{\alpha}_1 = \cdots = \boldsymbol{\alpha}_d$ learned simultaneously. The solution is given as

$$\widehat{\boldsymbol{\alpha}'} = \arg\min_{\boldsymbol{\alpha}'} \left[\boldsymbol{\alpha}'^\top \sum_{j=1}^{d} \boldsymbol{G}_j \boldsymbol{\alpha}' + 2 \sum_{j=1}^{d} \boldsymbol{h}_j^\top \boldsymbol{\alpha}' + \lambda \|\boldsymbol{\alpha}'\|^2 \right]$$
$$= -\left(\sum_{j=1}^{d} \boldsymbol{G}_j + \lambda \boldsymbol{I}_b \right)^{-1} \sum_{j=1}^{d} \boldsymbol{h}_j,$$

where $\lambda \geq 0$ is the ℓ_2-regularization parameter. This method agrees with MT-LSLDG at the limit $\gamma \to \infty$.

In all the methods, we set the number of basis functions as $b = \min\{50, n\}$, and randomly choose the kernel centers $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_b$ uniformly from training samples $\{\boldsymbol{x}_i\}_{i=1}^n$ without replacement. For hyper-parameters, we use the common ℓ_2-regularization parameter λ and bandwidth parameter σ among all the dimensions, i.e., $\lambda_1 = \cdots = \lambda_d = \lambda$ and $\sigma_1 = \cdots = \sigma_d = \sigma$. We also set all the similarity parameters as $\gamma_{j,j'} = 1$, which assumes that all dimensions are equally related to each other.

In order to examine whether this assumption is reasonable, we experimentally compare MT-LSLDG with this assumption and that with the similarity parameter tuning (Algorithhm 1) in Section 3.5.2.

3.5.2 Artificial data

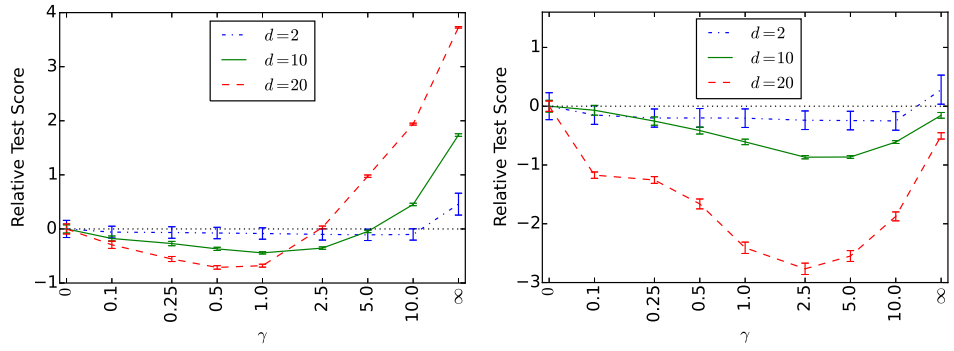
We conduct numerical experiments on artificial data to investigate the basic behavior of MT-LSLDG. As data density $p(\boldsymbol{x})$, we consider the following two cases:

- Single Gaussian: The d-dimensional Gaussian density whose mean is $\mathbf{0}$ and whose covariance matrix is the diagonal matrix with the first half of the diagonal elements are 1 and the others are 5.
- Double Gaussian: A mixture of two d-dimensional Gaussian densities with mean zero and $(5, 0, \ldots, 0)^\top$ and identity covariance matrix. The mixing coefficients are $1/2$.

The dimensionality d and sample size n are specified later. First, we investigate whether MT-LSLDG improves the estimation accuracy of LSLDG at appropriate γ. We prepare datasets with different dimensionalities $d = 2, 10, 20$ and sample sizes $n = 10, 30, 50$. MT-LSLDG is applied to the datasets at each $\gamma \in \{0, 0.1, 0.25, 0.5, 1, 2.5, 5, 10, \infty\}$. The Gaussian bandwidth σ and the ℓ_2-regularization parameter λ are chosen by 5-fold cross-validation as described in Section 3.4 from the candidate lists $\{10^{-1}, 10^{-0.25}, 10^{0.5}, 10^{1.25}, 10^2\}$ and $\{10^{-3}, 10^{-2}, 10^{-1}\}$, respectively. The solution of MT-LSLDG is computed analytically as in (3.9).

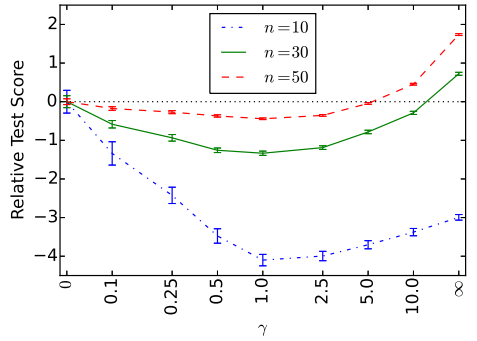
The results are plotted in Figure 3.2. In the figure, the relative test score is defined as the test score from which the test score of S-LSLDG is subtracted, and thus negative relative scores indicate that MT-LSLDG improved the performance of S-LSLDG. When $d = 2$, MT-LSLDG does not improve the performance for any γ values (Figure 3.2(a) and 3.2(b)). However, for higher-dimensional data, the performance is improved at appropriate γ values (e.g., $\gamma = 0.5$ for $d = 20$ in Figure 3.2(a) and $\gamma = 2.5$ for $d = 20$ in Figure 3.2(b)). Similar improvement is observed also for smaller sample size (e.g., $n = 10$ and $n = 30$) in Figure 3.2(c) and Figure 3.2(d).

These results confirm that MT-LSLDG improves the performance of S-LSLDG at an appropriate γ value when data is relatively high-dimensional and the sample size is small. Since such γ is usually unknown in advance, we need to find a reasonable value in practice.

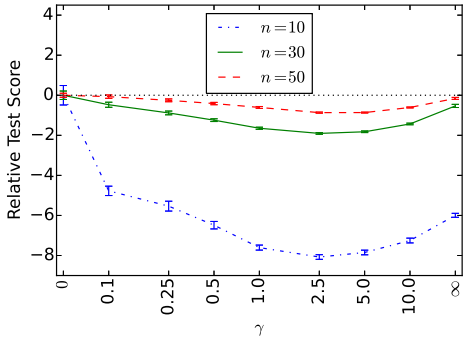


(A) Single Gaussian ($n = 30$)

(B) Double Gaussian ($n = 30$)



(C) Single Gaussian ($d = 10$)



(D) Double Gaussian ($d = 10$)

Figure 3.2: Average (and standard errors) of *relative test scores* over 100 runs. The relative test scores refer to test scores from which the test score of S-LSLDG is subtracted. The black dotted lines indicate the relative score zero.

Next, we investigate whether an appropriate γ value can be chosen by cross-validation. In this experiment, the cross-validation method in Section 3.4.4 is performed to choose γ as well. The candidates of γ is $\{0, 0.1, 0.25, 0.5, 1, 2.5, 5, 10, \infty\}$. The other experimental settings such as the data generation and all the LSLDGs are the same as in the last experiment except that we also run MT-LSLDG with similarity parameter tuning by Algorithm 1 with cross-validated α. The candidate list for α is $\{10^{-1}, 10^{-1/4}, 10^{2/4}, 10^{5/4}, 10^{2}\}$.

Table 3.1 shows that MT-LSLDG improves the performance especially when the dimensionality of data is relatively high and the sample size is small. These results indicate that the proposed cross-validation method allows us to choose a reasonable γ value.

In most cases of the experiments, MT-LSLDG with $\gamma_{j,j'} = 1$ gives estimation accuracy comparable to that with the similarity parameter tuning procedure. In the remaining experiments, we use cross-validated γ and the fixed $\gamma_{j,j'} = 1$.

3.5.3 Benchmark data

In this section, we demonstrate the usefulness of MT-LSLDG in gradient estimation on various benchmark datasets. This experiment uses some IDA benchmark datasets (Rätsch

Table 3.1: Averages (and standard errors) of test scores on the artificial data with cross-validation over 100 runs. MT-LSLDG-T in the table refers to MT-LSLDG with similarity parameter tuning by Algorithm 1. In each row, the best and comparable to the best scores in terms of paired t-test with significance level 5% are emphasized in bold face.

Density	n	MT-LSLDLG	MT-LSLDG-T	S-LSLDG	C-LSLDG
Single Gaussian	10	**−2.87** (0.22)	−2.33 (0.37)	0.37 (0.31)	**−2.58** (0.07)
	30	**−5.34** (0.04)	**−5.38** (0.02)	−4.97 (0.08)	−3.29 (0.03)
	50	**−5.63** (0.02)	**−5.64** (0.01)	−5.55 (0.02)	−4.13 (0.04)
Double Gaussian	10	**−6.83** (0.14)	**−6.80** (0.17)	1.01 (0.54)	−5.02 (0.12)
	30	**−8.45** (0.03)	**−8.45** (0.07)	−7.63 (0.10)	−7.84 (0.04)
	50	−8.67 (0.02)	**−8.71** (0.03)	−8.29 (0.10)	−8.48 (0.02)

Density	d	MT-LSLDG	MT-LSLDG-T	S-LSLDG	C-LSLDG
Single Gaussian	2	**0.20** (0.19)	0.21 (0.15)	**−0.11** (0.15)	**0.09** (0.15)
	10	**−5.34** (0.04)	**−5.38** (0.02)	−4.97 (0.08)	−3.29 (0.03)
	20	**−10.77** (0.03)	**−10.76** (0.04)	−9.98 (0.13)	−6.39 (0.01)
Double Gaussian	2	**0.54** (0.22)	**0.51** (0.25)	**0.50** (0.27)	**0.19** (0.22)
	10	**−8.45** (0.03)	**−8.45** (0.07)	−7.63 (0.10)	−7.84 (0.04)
	20	**−16.88** (0.14)	**−17.06** (0.125)	−14.90 (0.10)	−15.26 (0.06)

et al., 2001) and UCI benchmark datasets (Catlett, 1991; Kaya et al., 2012; Lichman, 2013; Lucas et al., 2013; Tüfekci, 2014). All the datasets are standardized in advance.

For MT-LSLDG, the hyper-parameters σ, λ and γ are chosen by cross-validation. The candidate lists are $\sigma \in \{0.1, 0.25, 0.5, 1, 2.5, 5, 10\}$, $\lambda \in \{10^{-5}, 10^{-4}, \ldots, 10^{-1}\}$ and $\gamma \in \{0, 10^{-5}, 10^{-4}, \ldots, 10^{1}, 10^{2}, \infty\}$, respectively. For S-LSLDG and C-LSLDG, the candidate lists of σ and λ are the same as MT-LSLDG. The solution of MT-LSLDG is computed by the BCD algorithm described in Section 3.4.5.2.

The results are presented in Table 3.2. MT-LSLDG significantly improves the performance of either S-LSLDG or C-LSLDG on most of the datasets.

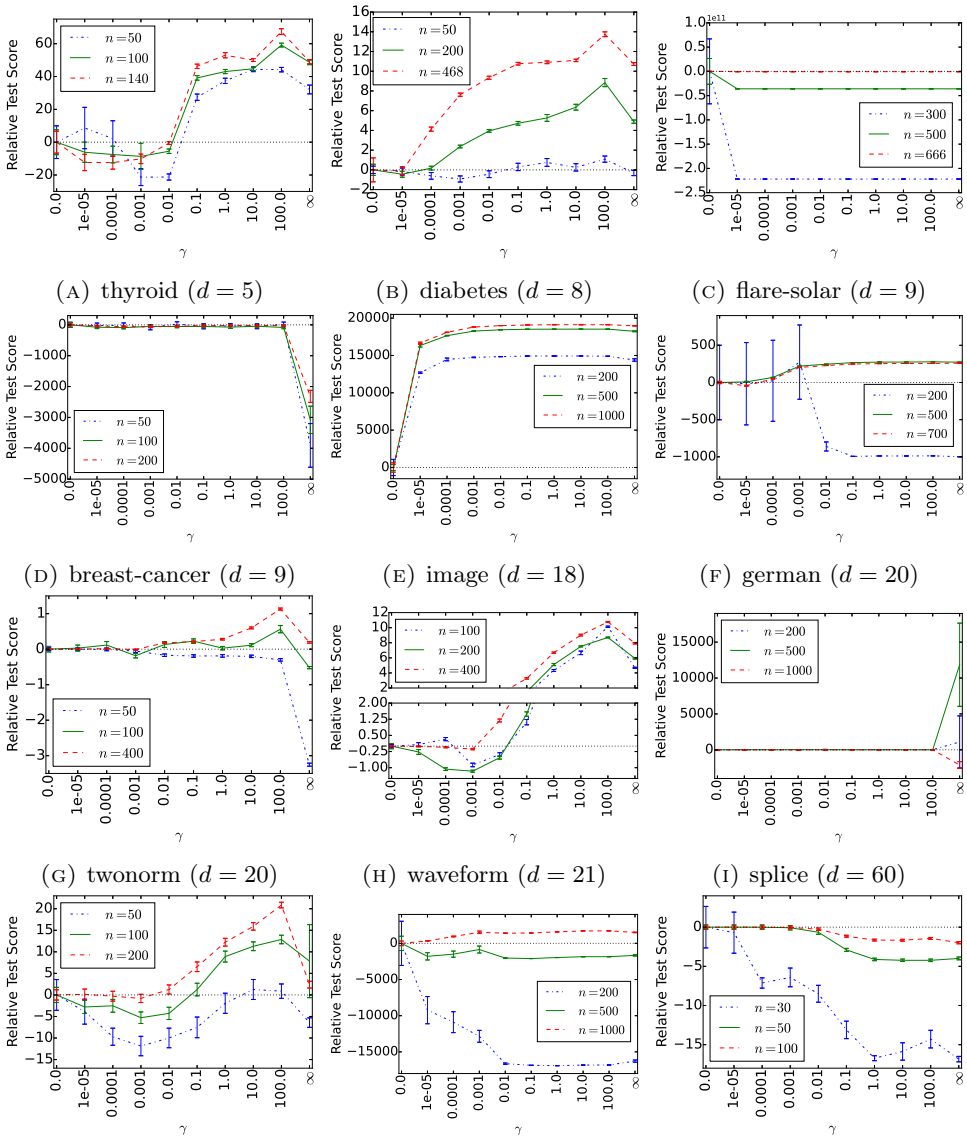
We also run MT-LSLDG for three different samples sizes n for each of the datasets with changing γ from 0 to ∞. Figure 3.3 summarizes the results of the experiments. The plots show that performance highly depends on γ, and that the best γ differs from one dataset to another, which means that it is very important to select a good γ by cross-validation. Also, we can see that the blue plot, which is that for the smallest n, shows the largest improvement over S-LSLDG in most of the datasets. This implies that MT-LSLDG is advantageous especially when the sample size is small.

3.6 Application to mode-seeking clustering

In this section, we apply MT-LSLDG to mode-seeking clustering and experimentally demonstrate its usefulness.

Table 3.2: Averages (and standard errors) of the test scores on the benchmark datasets. In each dataset, the best and comparable to the best scores in terms of paired t-test with significance level 5% are emphasized in bold face. The number of trials is 20 for the image and splice dataset, and is 100 for the other datasets.

Dataset (d, n)	MT-LSLDG	S-LSLDG	C-LSLDG
thyroid (5, 140)	$\mathbf{-1.076 \times 10^2}$	$\mathbf{-1.083 \times 10^2}$	-5.149×10
	(0.066×10^2)	(0.065×10^2)	(0.012×10)
CCPP (5, 200)	$\mathbf{-3.661 \times 10}$	$\mathbf{-3.585 \times 10}$	-3.232×10
	(0.120×10)	(0.125×10)	(0.187×10)
diabetes (8, 468)	$\mathbf{-2.240 \times 10}$	$\mathbf{-2.211 \times 10}$	-1.510×10
	(0.029×10)	(0.034×10)	(0.008×10)
flare-solar (9, 666)	$\mathbf{-1.341 \times 10^7}$	6.626×10^8	$\mathbf{-1.342 \times 10^7}$
	(0.021×10^7)	(3.251×10^8)	(0.021×10^7)
breast-cancer (9, 200)	$\mathbf{-2.535 \times 10^3}$	-2.169×10^2	$\mathbf{-2.535 \times 10^3}$
	(0.195×10^3)	(0.294×10^2)	(0.195×10^3)
shuttle (9, 1000)	$\mathbf{-2.664 \times 10^3}$	$\mathbf{-2.974 \times 10^3}$	-1.063×10^3
	(0.321×10^3)	(0.109×10^3)	(0.038×10^3)
image (18, 1300)	$\mathbf{-2.993 \times 10^3}$	1.020×10^4	$\mathbf{-3.289 \times 10^3}$
	(0.541×10^3)	(1.246×10^4)	(0.027×10^3)
popfailures (18, 50)	$\mathbf{-2.110 \times 10^2}$	-2.067×10^2	-2.108×10^2
	(0.002×10^2)	(0.003×10^2)	(0.002×10^2)
german	$\mathbf{-3.042 \times 10^2}$	$\mathbf{-2.960 \times 10^2}$	-1.999×10
	(0.140×10^2)	(0.092×10^2)	(0.373×10)
twonorm (20, 400)	$\mathbf{-2.233 \times 10}$	-2.232×10	-2.213×10
	(0.001×10)	(0.001×10)	(0.002×10)
waveform (21, 400)	$\mathbf{-4.332 \times 10}$	-4.321×10	-3.526×10
	(0.003×10)	(0.002×10)	(0.010×10)
splice (60, 1000)	$\mathbf{-2.484 \times 10^3}$	-6.027×10	$\mathbf{-2.382 \times 10^3}$
	(0.379×10^3)	(0.345×10)	(0.393×10^3)



(A) thyroid ($d = 5$) (B) diabetes ($d = 8$) (C) flare-solar ($d = 9$)

(D) breast-cancer ($d = 9$) (E) image ($d = 18$) (F) german ($d = 20$)

(G) twonorm ($d = 20$) (H) waveform ($d = 21$) (I) splice ($d = 60$)

(J) CCPP ($d = 5$) (K) shuttle ($d = 9$) (L) popfailures ($d = 18$)

Figure 3.3: Average (and standard errors) of relative test scores on the IDA datasets and the other real datasets. The relative test scores refer to test scores from which the test score of S-LSLDG is subtracted. The black dotted lines indicate the relative score zero.

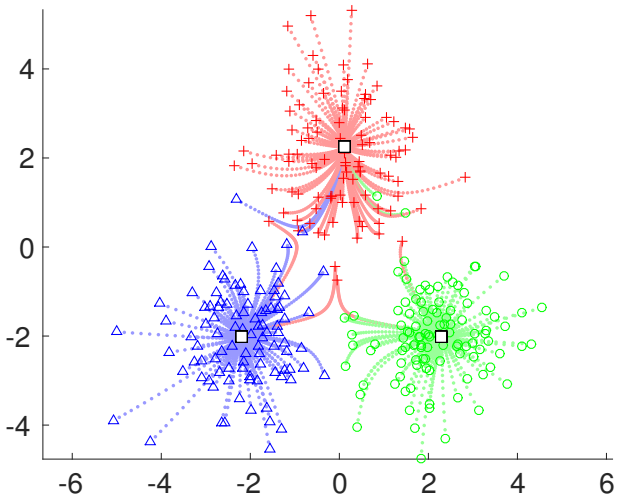


Figure 3.4: Transition of data points during a mode-seeking process. Data samples are drawn from a mixture of Gaussians, and the data points sampled from the same Gaussian component are specified by the same color (red, green, or blue) and marker (plus symbol, circle, or triangle). White squares indicate the points to which data points converged.

3.6.1 Mode-seeking clustering

A practical application of log-density gradient estimation is *mode-seeking clustering* (Cheng, 1995; Comaniciu and Meer, 2002; Fukunaga and Hostetler, 1975; Sasaki et al., 2014). Mode-seeking clustering methods update each data point toward a nearby mode by gradient ascent, and assign the same clustering label to the data points which converged to the same mode (Figure 3.4). Their notable advantage is that we need not specify the number of clusters in advance. Mode-seeking clustering has been successfully applied to a variety of real world problems such as object tracking (Comaniciu et al., 2000), image segmentation (Comaniciu and Meer, 2002; Sasaki et al., 2014), and line edge detection in images (Bandera et al., 2006).

In mode-seeking, the essential ingredient is the gradient of the data density. To estimate the gradients, *mean shift clustering* (Cheng, 1995; Comaniciu and Meer, 2002; Fukunaga and Hostetler, 1975), which is one of the most popular mode-seeking clustering methods, employs the two-step method of first estimating the data density by kernel density estimation and then taking its gradient. However, as we mentioned earlier, this two-step method does not work well since accurately estimating the density does not necessarily lead to an accurate estimate of the gradient.

In order to overcome this problem, *LSLDG clustering* (Sasaki et al., 2014) adopted LSLDG instead of the two-step method. Sasaki et al. (2014) also provided a practically useful fixed-point algorithm for mode-seeking as in mean shift clustering (Cheng, 1995): When the partial derivative of a vector of Gaussian kernels $\boldsymbol{\psi}_j(\boldsymbol{x}) = \partial_j \boldsymbol{\phi}(\boldsymbol{x})$ is used as the

vector of basis functions, the model $g_j(\boldsymbol{x}) = \widehat{\boldsymbol{\alpha}}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x})$ can be transformed as

$$\begin{aligned}\widehat{g}_j(\boldsymbol{x}) &= \sum_{k=1}^{b} \widehat{\alpha}_j^{(k)} \frac{c_k^{(j)} - x^{(j)}}{\sigma^2} \phi^{(k)}(\boldsymbol{x}) \\ &= \frac{1}{\sigma^2} \sum_{k=1}^{b} \widehat{\alpha}_j^{(k)} c_k^{(j)} \phi^{(k)}(\boldsymbol{x}) - \frac{x^{(j)}}{\sigma^2} \sum_{k=1}^{b} \widehat{\alpha}_j^{(k)} \phi^{(k)}(\boldsymbol{x}) \\ &= \left[\frac{1}{\sigma^2} \sum_{k=1}^{b} \widehat{\alpha}_j^{(k)} \phi^{(k)}(\boldsymbol{x}) \right] \left[\frac{\sum_{k'=1}^{b} \widehat{\alpha}_j^{(k')} c_{k'}^{(j)} \phi^{(k')}(\boldsymbol{x})}{\sum_{k'=1}^{b} \widehat{\alpha}_j^{(k')} \phi^{(k')}(\boldsymbol{x})} - x^{(j)} \right],\end{aligned}$$

where we assume that $\frac{1}{\sigma^2} \sum_{k=1}^{b} \widehat{\alpha}_j^{(k)} \phi^{(k)}(\boldsymbol{x})$ is nonzero. Setting $\widehat{g}_j(\boldsymbol{x})$ to zero yields a fixed-point update formula as

$$x^{(j)} \leftarrow \frac{\sum_{k'=1}^{b} \widehat{\alpha}_j^{(k')} c_{k'}^{(j)} \phi^{(k')}(\boldsymbol{x})}{\sum_{k'=1}^{b} \widehat{\alpha}_j^{(k')} \phi^{(k')}(\boldsymbol{x})}.$$

It has been experimentally shown that LSLDG clustering performs significantly better than mean-shift clustering (Sasaki et al., 2014).

Here, we apply MT-LSLDG to LSLDG clustering and investigate if the performance is improved in mode-seeking clustering as well for relatively high-dimensional data.

3.6.2 Experiments

Next, we conduct numerical experiments for mode-seeking clustering.

3.6.2.1 Experimental setting

We apply the following four clustering methods to various datasets:

- MT-LSLDGC: LSLDG clustering with MT-LSLDG.
- S-LSLDGC: LSLDG clustering with S-LSLDG (Sasaki et al., 2014).
- C-LSLDGC: LSLDG clustering with C-LSLDG (Sasaki et al., 2014).
- Mean-shift: mean shift clustering (Comaniciu and Meer, 2002).

For MTL-, S-, and C-LSLDG, all the hyper-parameters are cross-validated as described in Section 3.4.4, and for mean-shift, log-likelihood cross-validation is used.

We evaluate the clustering performance by the *adjusted Rand index* (ARI) (Hubert and Arabie, 1985). ARI gives one to the perfect clustering assignment and zero on average to a random clustering assignment. A larger ARI value means a better clustering result.

3.6.2.2 Artificial data

First, we conduct experiments on artificial data. The density of the artificial data is a mixture of three d-dimensional Gaussian densities with means $(0, 2, 0, \ldots, 0)$, $(-2, -2, 0, \ldots, 0)$, and

Table 3.3: Averages (and standard errors) of ARIs on artificial data. In each row, the best and comparable to the best ARI in terms of unpaired t-test with significance level 5% is emphasized in bold face. The number of trials is 100.

d	MT-LSLDGC	S-LSLDGC	C-LSLDGC	Mean-shift
2	**0.992** (0.035)	**0.973** (0.125)	**0.992** (0.036)	**0.984** (0.044)
10	**0.993** (0.004)	**0.994** (0.003)	**0.994** (0.004)	0.042 (0.022)
15	**0.983** (0.023)	**0.982** (0.054)	0.877 (0.217)	0.000 (0.000)
20	**0.827** (0.190)	0.586 (0.208)	0.716 (0.352)	0.036 (0.037)

$(2, -2, 0, \ldots, 0)$, covariance matrices $\frac{1}{\sqrt{2\pi}}\boldsymbol{I}_d$, and mixing coefficients 0.4,0.3,0.3. The candidate lists of the hyper-parameters are the following: $\sigma \in \{10^{-1}, 10^{-7/9}, 10^{-5/9}, \ldots, 10^{5/9}, 10^{7/9}, 10^{1}\}$, $\lambda \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and, $\gamma \in \{0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, \infty\}$.

The results are shown in Table 3.3. We can see that MT-LSLDGC performs well especially for the largest dimensionality $d = 20$.

3.6.2.3 Real data

Next, we perform clustering on real data. The following three datasets are used:

- Accelerometry data: 5-dimensional data used in (Hachiya et al., 2012) for human activity recognition extracted from mobile sensing data available from `http://alkan.mns.kyutech.ac.jp/web/data`. The number of classes is 3. In each run of experiment, we use randomly chosen 100 samples from each class. The total number of samples is 300.

- Vowel data: 10-dimensional data of recorded British English vowel sounds available from `https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Vowel+Recognition+-+Deterding+Data)`. The number of classes is 11 In each run of experiment, we use randomly chosen 500.

- Sat-image data: 36-dimensional multi-spectral satellite image available from `https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)`. The number of classes is 6. In each run of experiment, we use randomly chosen 2000 samples.

- Speech data: 50-dimensional voice data by two French speakers (Sugiyama et al., 2014). The number of classes is 2. In each run of experiment, we use randomly chosen 200 samples from each class. The total number of samples is 400.

For MT-LSLDG, the hyper-parameters are cross-validated using the candidates, $\sigma \in \{10^{-1}, 10^{-6/9}, 10^{-3/9}, \ldots, 10^{12/9}, 10^{15/9}, 10^{2}\}$, $\lambda \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ and $\gamma \in \{10^{-5}, 10^{-4}, \ldots, 10^{1}, 10^{2}\}$, except that we use relatively small candidate lists $\sigma \in \{0.5, 1, 2.5, 5, 10\}$, $\lambda \in \{0.003, 0.01, 0.1, 1\}$ and $\gamma \in \{0.1, 1, 10\}$ for the speech data since it has large dimensionality and optimization is computationally expensive. For S- and C-LSLDG, we used the same candidates of MT-LSLDG for σ and λ. For mean shift clustering, the Gaussian kernel is employed in KDE, and the bandwidth parameter in the kernel is selected

Table 3.4: Averages (and standard errors) of ARIs on real data. In each row, the best and comparable to the best ARI in terms of paired t-test with significance level 5% is emphasized in bold face. The number of trials is 100 for the accelerometry data and the sat-image data, and is 20 for the speech data.

dataset (d, n)	MT-LSLDGC	S-LSLDGC	C-LSLDGC	Mean-shift
accelerometry (5, 300)	0.40 (0.01)	**0.53** (0.02)	0.24 (0.01)	0.26 (0.04)
vowel (10, 500)	**0.15** (0.00)	**0.15** (0.00)	**0.15** (0.00)	0.04 (0.00)
sat-image (36, 2000)	**0.48** (0.00)	0.43 (0.01)	0.35 (0.00)	0.00 (0.00)
speech (50, 400)	**0.17** (0.02)	0.00 (0.00)	0.15 (0.01)	0.00 (0.00)

by 5-fold cross-validation with respect to the log-likelihood of the density estimate from the same candidates of MT-LSLDG for σ.

The results are shown in Table 3.4. For the accelerometry data whose dimensionality is only five, S-LSLDGC gives the best performance and MT-LSLDGC does not improve the performance, although MT-LSLDGC performs better than C-LSLDGC.

On the other hand, for the higher-dimensional dataset, the vowel data, the sat-image data, and the speech data, the performance of MT-LSLDGC is the best or comparable to the best. These results indicate that MT-LSLDG is a promising method in mode-seeking clustering especially when the dimensionality of data is relatively large.

3.7 Conclusion

We proposed a multi-task log-density gradient estimator in order to improve the existing estimator in higher-dimensional cases. Our fundamental idea is to exploit the relatedness inhering in the partial derivatives through regularized multi-task learning. As a result, we experimentally confirmed that our method significantly improves the accuracy of log-density gradient estimation. Finally, we demonstrated its usefulness of the proposed log-density gradient estimator in mode-seeking clustering.

Although fixing the similarity parameters $\gamma_{j,j'}$ to be 1 worked reasonably well in our experiments, carefully tuning them may further improve the estimation accuracy. A good practice may be to use the heuristic procedure given in Algorithm 1, whose properties have yet to be analyzed. Another way is to use Bayesian optimization techniques such as the Gaussian process approaches studied in Bergstra et al. (2011) and Snoek et al. (2012), which have been experimentally shown to be reasonably fast even in large-scale hyper-parameter tuning tasks.

As log-density gradient is a vector-valued function learning problem, one may consider applying kernel-based methods for such problems (Caponnetto et al., 2008; Micchelli and Pontil, 2005a). Micchelli and Pontil (2005a) showed a representer theorem for a wide class of optimizations in a reproducing kernel Hilbert space of vector-valued functions whose objective functional depends only on outputs of the function to be optimized. However, it may not be possible to directly employ those methods in the LSLDG framework since the

LSLDG objective functional also depends on the gradients besides outputs of the function. It is an important open question whether LSLDG also admits a similar representer theorem or not.

Log-density gradient estimation would be useful in a measure for non-Gaussianity (Huber, 1985) and other further fundamental statistical topics (Singh, 1977). In the future work, we will investigate the performance of our proposed method in these topics.

Chapter 4

Multi-Task Principal Component Analysis

Principal Component Analysis (PCA) is a canonical and well-studied tool for unsupervised dimensionality reduction. However, when only a limited amount of data are available, the poor quality of the covariance estimate at its core may compromise its performance. We mitigate this issue when there are multiple similar PCA tasks to be solved at hand by casting the PCA tasks into a multi-task framework. We propose a formulation of the multi-task PCA problem using a novel multi-task regularization. This regularization is based on a distance between projection matrices, and the whole problem is solved as an optimization problem defined on the Riemannian manifold consisting of projection matrices. We experimentally demonstrate the usefulness of our approach as pre-processing for EEG signals.

4.1 Introduction

Principal Component Analysis (PCA) (Hotelling, 1933; Joliffe, 1986; Pearson, 1901) is a data preprocessing technique widely used in data processing and is a prominent dimensionality reduction technique in machine learning.

In a few words, PCA seeks an accurate low-dimensional approximation to high-dimensional data. To do so, PCA finds an orthogonal projection of the data to a low-dimensional subspace while preserving as much variance as possible, or equivalently while minimizing the projection error (see Bishop (2006a, Chap 12)).

In practice, this boils down to a simple eigenvalue problem involving the empirical covariance of the input training samples. Its simplicity and efficiency allowed the extensions to several variants of PCA over the course of time, ranging from non-linear extensions (Schölkopf et al., 1997; Vincent et al., 2010) to sparse (Zou et al., 2006) or supervised extensions (De Bie et al., 2005). PCA has been studied also from the point of view of subspace tracking in order to efficiently cope with non-stationary data streams (Badeau et al., 2008; Balzano et al., 2010), where the emphasis is put on efficiently updating the principal subspace while maintaining the orthonormality constraint. In a related setup, it has also been studied from

the online learning point of view (Warmuth and Kuzmin, 2007) in order to derive bounds on the projection error.

When we want to project data onto a one-dimensional subspace, the PCA can be solved by extracting the dominant eigenvector of the sample covariance matrix of the input data. Multi-dimensional PCA can be performed by iteratively solving one-dimensional problems in a deflation scheme. However, this problem can also be solved at once by optimizing a generalization of the one-dimensional cost under orthonormality constraints or by optimizing on a Riemannian manifold[*1] involving orthogonal matrices (Absil et al., 2009; Edelman et al., 1998). When such a cost is optimized, the solution may not exactly diagonalize the covariance matrix but will have the same span as the leading eigenvectors.

As any machine learning method, the quality of the solution obtained by the PCA is greatly affected in practice by the quality of the input, which in this case is the sample covariance. Being based on the minimization of a least-squares cost, the quality of the covariance estimator is particularly affected by outliers. Hence, in order to overcome this situation, several robust versions of the PCA have been proposed for dealing with noisy data and outliers. Those approaches either rely on multivariate trimming of the samples (Devlin et al., 1981) or on a cost function giving less influence to outliers (Candès et al., 2011).

However, in a context where only a limited amount of data are available, the covariance matrix may not be accurately estimated, nor the robust approaches are not adapted. If such a situation happens to several related PCA tasks, one straightforward approach consists of finding a common principal subspace to all the tasks. As studied in Wang et al. (2011), it boils down to applying a single PCA over all the data or to finding a subspace approximating all the covariance matrices. This latter formulation makes the problem close to the Approximate Joint Diagonalization (AJD) encountered in the Signal Processing community (Cardoso and Souloumiac, 1996; Flury and Gautschi, 1986). However, these approaches do not take into account the heterogeneity of the tasks.

On the other hand, in this context of data scarcity, as the covariance estimator is not reliable, independently solving a PCA task for every dataset would fail. Hence, we need a trade-off between 1) the flexible approach of independently and separately solving the PCA tasks and 2) the restrictive approach of finding the single, common subspace for all the data from the tasks at once. To do so, we propose to cast the PCA tasks into the Multi-task Learning (MTL) framework (Argyriou et al., 2008b; Caruana, 1998; Evgeniou and Pontil, 2004b; Zhang and Yeung, 2011). In this setup, every task corresponds to finding a low-rank orthogonal projection of each dataset (maximizing the retained information). We solve those tasks simultaneously with a multi-task regularization term that makes those projections similar to each other. As we focus on the multi-dimensional case, we formulate our problem as an optimization problem over a Riemannian matrix manifold. Our multi-task regularization is designed based on a distance metric intrinsic to the geometry of this space.

[*1] A Riemannian manifold is a smoothly curved non-Euclidean space with additional structures such as a set of linear local approximations, i.e. the tangent spaces, that are equipped with an inner product (Absil et al., 2009).

Eigenvalue problems being a classical tool of machine learning (De Bie et al., 2005), their study in the multi-task framework has naturally been proposed. Recently, such an approach has been developed in Wang et al. (2016). This approach studied the generalized eigenvalue problems and only extracted the leading eigenvector by casting the problem into a multi-task dictionary learning problem. In essence, our contribution in our work is different from this previous work. While they focused on one-dimensional generalized eigenvalue problems, our focus is on standard PCA problems, but our method is able to extract directly a dominant *subspace* (i.e. the span of a set of leading eigenvectors) without having to resort to any multi-stage deflation scheme. As exposed in this chapter, we propose a simple and elegant MTL formulation relying on a novel regularization.

The use of a multi-task methodology has been advocated in challenging applications such as Brain Computer Interfaces (BCI) where it is difficult to collect data from each task but the tasks are related to each other (Devlaminck et al., 2011; Samek et al., 2013). In this chapter, we provide some promising results in this difficult application. In order to analyze the behavior of our approach, we also apply it on synthetic data.

To summarize, the key contributions of our work are twofold: First and foremost, we formulate the problem of dominant subspace extraction for multi-task variance maximization on a matrix manifold. As a result, it makes it possible to solve at once several related PCA problems of fixed dimensionality. Secondly, we propose a relevant regularization (having an interpretation from the viewpoint of the Riemannian geometry) for this multi-task problem. Then, the problem is naturally formulated as an optimization problem over a Riemannian matrix manifold. Through experiments on synthetic data and a signal processing application, we demonstrate the efficacy of our proposed dimensionality reduction method.

4.2 Multi-task Variance Maximization

In this section, we define the problem of multi-task variance maximization and then present our proposed method.

4.2.1 Problem Setup

This problem being defined as a collection of instances of single-task variance maximization, we first start by introducing the single-task version of variance maximization.

For any random data variable $\boldsymbol{x} \in \mathbb{R}^d$ following some unknown probability density $p(\boldsymbol{x})$, the goal of variance maximization is to estimate the k-dimensional subspace ($k < d$; we assume k is known and fixed) on which the projected point of $\boldsymbol{x}$ has the maximum variance, from i.i.d. samples $\{\boldsymbol{x}_i\}_{i=1}^n$ drawn from $p(\boldsymbol{x})$.

For any matrix $\boldsymbol{M} \in \mathbb{R}^{d\times k}$, we denote the span of the columns of $\boldsymbol{M}$ by $\mathrm{Span}(\boldsymbol{M})$. For any k-dimensional subspace S and any orthogonal matrix $\boldsymbol{U} \in \mathbb{R}^{d\times k}$, we say that $\boldsymbol{U}$ is an *orthogonal basis matrix* of S if $\mathrm{Span}(\boldsymbol{U}) = S$. Any d-by-k orthogonal matrix determines a unique subspace as an orthogonal basis matrix while there are infinitely many orthogonal

basis matrices for any given subspace with dimensionality $d \geq 2$. Since the orthogonal projection onto any subspace S is given as $\mathbb{R}^d \ni \boldsymbol{x} \mapsto \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x} \in S$ using any basis matrix $\boldsymbol{U}$ of S, an orthogonal basis matrix $\boldsymbol{U}^*$ of the optimal subspace S^* is obtained as a solution to the following problem:

$$\boldsymbol{U}^* = \operatorname*{argmax}_{\boldsymbol{U} \in \mathbb{R}^{d \times k}:\ \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k} \mathop{\mathbf{E}}_{\boldsymbol{x} \sim p(\boldsymbol{x})} \left[\| \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{\mu} \|^2 \right] \tag{4.1}$$

$$= \operatorname*{argmax}_{\boldsymbol{U} \in \mathbb{R}^{d \times k}:\ \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k} \operatorname{Tr}(\boldsymbol{U}^\top \boldsymbol{C} \boldsymbol{U}), \tag{4.2}$$

where $\boldsymbol{\mu} = \mathbf{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})}[\boldsymbol{x}]$ is the population mean of $\boldsymbol{x}$, $\boldsymbol{C} = \mathbf{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^\top]$ is the population covariance of $\boldsymbol{x}$, and $\boldsymbol{I}_k$ is the k-by-k identity matrix. Again, $\boldsymbol{U}^*$ is not uniquely determined because the objective function is invariant under orthogonal transformations since $\operatorname{Tr}((\boldsymbol{U}\boldsymbol{O})^\top \boldsymbol{C} (\boldsymbol{U}\boldsymbol{O})) = \operatorname{Tr}(\boldsymbol{U}^\top \boldsymbol{C}\boldsymbol{U}\boldsymbol{O}\boldsymbol{O}^\top) = \operatorname{Tr}(\boldsymbol{U}^\top \boldsymbol{C}\boldsymbol{U})$ for any orthogonal matrix $\boldsymbol{O} \in \mathbb{R}^{k \times k}$. As made clear later in this chapter, in order to deal with the orthogonality constraint as well as with this invariance to rotations, we will use Grassmann manifolds for the formulation of the problem (Edelman et al., 1998).

In *multi-task variance maximization*, which is the main subject of this chapter, we have multiple different instances of variance maximization. We call such instances as *tasks*. More specifically, given T sets of i.i.d. samples $\{\boldsymbol{x}_{t,i}\}_{i=1}^{n_t}$, $t = 1, \ldots, T$, following underlying probability densities $p_1(\boldsymbol{x}_1), \ldots, p_T(\boldsymbol{x}_T)$ respectively, we are required to estimate the optimal k-dimensional subspaces, whose basis matrices $\boldsymbol{U}_t^*$, $t = 1, \ldots, T$, are given by

$$\boldsymbol{U}_t^* = \operatorname*{argmax}_{\boldsymbol{U}_t \in \mathbb{R}^{d \times k}:\ \boldsymbol{U}_t^\top \boldsymbol{U}_t = \boldsymbol{I}_k} \operatorname{Tr}(\boldsymbol{U}_t^\top \boldsymbol{C}_t \boldsymbol{U}_t), \tag{4.3}$$

where $\boldsymbol{\mu}_t = \mathbf{E}_{\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t)}[\boldsymbol{x}_t]$, and $\boldsymbol{C}_t = \mathbf{E}_{\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t)}[(\boldsymbol{x}_t - \boldsymbol{\mu}_t)(\boldsymbol{x}_t - \boldsymbol{\mu}_t)^\top]$.

4.2.2 Principal Component Analysis

In many applications, the population covariance matrix $\boldsymbol{C}_t$ is often unknown, and the objective function of Eq. (4.3) cannot be directly evaluated. A common way to alleviate this is to resort to the *sample covariance matrix* defined by $\widehat{\boldsymbol{C}}_t = \frac{1}{n_t - 1} \sum_{i=1}^{n_t} (\boldsymbol{x}_{t,i} - \widehat{\boldsymbol{\mu}}_t)(\boldsymbol{x}_{t,i} - \widehat{\boldsymbol{\mu}}_t)^\top$ with $\widehat{\boldsymbol{\mu}}_t = \frac{1}{n_t} \sum_{i=1}^{n_t} \boldsymbol{x}_{t,i}$ to approximate the objective function as $\operatorname{Tr}(\boldsymbol{U}_t^\top \boldsymbol{C}_t \boldsymbol{U}_t) \approx \operatorname{Tr}(\boldsymbol{U}_t^\top \widehat{\boldsymbol{C}}_t \boldsymbol{U}_t)$.

In the case of the single task learning setup (i.e. $T = 1$), the method of solving such an approximated problem is widely known as Principal Component Analysis (PCA) (see, e.g., Joliffe (1986)) and can be solved by taking the leading k orthonormal eigenvectors of $\widehat{\boldsymbol{C}}_t$. PCA and its variants have been proven to be useful in many applications such as model reduction in control theory (Moore, 1981) and denoising for image processing (Zhang et al., 2010). In our multi-task setting, we refer to the method of applying PCA to every task independently as *Independent PCA (I-PCA)* and the method of applying it to the union of the datasets from all the tasks as *Common PCA (C-PCA)*. The notable difference between

these two methods is that C-PCA gives the same subspace for all the tasks whereas I-PCA could give completely different subspaces for different tasks.

I-PCA may provide good estimates of the optimal subspaces when sufficiently many data samples are available, but when we have only scarce data samples, the solutions $\widehat{\boldsymbol{U}}_t$ to the problem in Eq. (4.3) may be badly affected by unreliable covariance estimation resulting in poor performance on unseen data. In fact, the solution is undetermined when the sample size n_t is less than the dimensionality k of the subspace.

A straightforward countermeasure to this data-scarcity problem is to adopt C-PCA in order to simply increase the sample size. However, this corresponds to assuming that all the tasks share the identical optimal solution, which may be unreasonable when the tasks have considerable heterogeneity.

The objective of this work is to improve the performance over both I-PCA and C-PCA when the tasks are different but related to each other in the sense that their optimal subspaces are similar to each other. In such a case, solving all the tasks simultaneously while sharing information with each other may improve performance. This strategy of jointly learning multiple tasks with taking the advantage of their relatedness is called *multi-task learning* and has been shown to work well in many other applications (Argyriou et al., 2008b; Caruana, 1998; Evgeniou and Pontil, 2004b; Jacob et al., 2009b; Zhang and Yeung, 2011).

4.2.3 Regularized Multi-task PCA

One of the most successful approaches to multi-task learning is the regularization approach (Argyriou et al., 2008b; Evgeniou and Pontil, 2004b; Jacob et al., 2009b). In this approach, the tasks maintain different learning parameters but they are simultaneously optimized with an appropriately designed regularization term which, e.g., makes the parameters close to each other or imposes similar sparsity patterns on them.

In this chapter, we propose a method based on this approach for solving multi-task variance maximization. In the proposed method, we directly search the space of subspaces instead of searching the space of orthogonal skinny matrices. More specifically, we solve the following optimization problem:

$$(\widehat{\boldsymbol{S}}_1, \ldots, \widehat{\boldsymbol{S}}_T) = \underset{\substack{(\boldsymbol{U}_1,\ldots,\boldsymbol{U}_T)\\ \in \mathrm{Gr}(d,k)^{\otimes T}}}{\operatorname{argmax}} \underbrace{\left[\frac{1}{2} \sum_{t\in[T]} \mathrm{Tr}(\boldsymbol{U}_t^\top \widehat{\boldsymbol{C}}_t \boldsymbol{U}_t) + \frac{\lambda}{4} \sum_{s,t\in[T]:\, s\neq t} \mathrm{Tr}(\boldsymbol{U}_s \boldsymbol{U}_s^\top \boldsymbol{U}_t \boldsymbol{U}_t^\top) \right]}_{J(\boldsymbol{U}_1,\ldots,\boldsymbol{U}_T)}, \quad (4.4)$$

where $\lambda > 0$ is a regularization parameter, $[T] = \{1, \ldots, T\}$, and $\mathrm{Gr}(d,k)^{\otimes T}$ denotes the product manifold consisting of T *Grassmann manifolds*. Each of those manifolds consists of all the k-dimensional linear subspaces of the d-dimensional Euclidean space $\mathbb{R}^d$[*2], and $\widehat{\boldsymbol{S}}_t$ is the estimate of the optimal subspace for task t. We call this method *Regularized MultiTask Principal Component Analysis (RMT-PCA)*. As we will see later, the objective function does

[*2]Note that a point X on this manifold can be represented by any orthonormal basis of $\mathbb{R}^{k\times d}$. The chosen orthonormal basis is called a representative of its subspace $\mathrm{Span}(X)$.

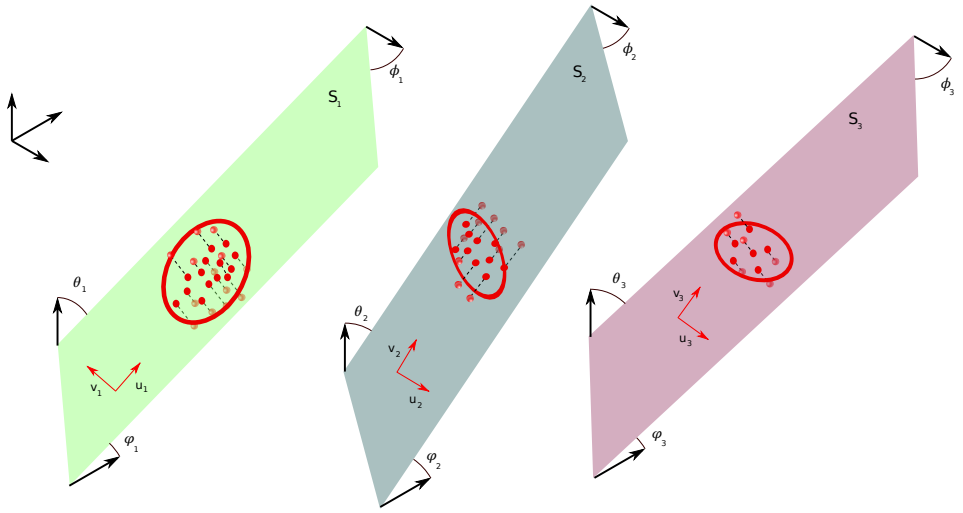


Figure 4.1: Illustration of the multi-task setup for the PCA problem. Few observations are available for every task of PCA, and we aim at extracting similar subspaces (hence being oriented according to similar angles). In this example, each subspace S_t is represented by a basis of two vectors u_t, v_t and the angles between the canonical basis and the subspaces are ϕ_t, θ_t, ψ_t.

not depend on the choice of the orthogonal basis matrices $\boldsymbol{U}_t$, $t = 1, \ldots, T$, and thus the optimization problem is well-defined on $\mathrm{Gr}(d,k)^{\otimes T}$.

Intuitively, we try to maximize the PCA objective function $\mathrm{Tr}(\boldsymbol{U}_t^\top \widehat{\boldsymbol{C}} \boldsymbol{U}_t)$ for every task t simultaneously while maximizing the similarity between the subspaces $\mathrm{Span}(\boldsymbol{U}_s)$ and $\mathrm{Span}(\boldsymbol{U}_t)$ quantified by $\mathrm{Tr}(\boldsymbol{U}_s \boldsymbol{U}_s^\top \boldsymbol{U}_t \boldsymbol{U}_t^\top)$ for every task pair (s,t) at the same time.

Figure 4.1 illustrates the idea of our multi-task PCA approach. In this example, three datasets of three-dimensional examples are observed. Those three datasets share similar (but slightly different) behaviors as their two-dimensional principal subspaces are close to be parallel. Hence, the overall objective is to find similar subspaces (i.e. having similar angles) expressing most of the variance of each dataset. This example shows the flexibility of our approach as it is immune to the choice of bases representing the subspaces.

Maximizing the term $\mathrm{Tr}(\boldsymbol{U}_s \boldsymbol{U}_s^\top \boldsymbol{U}_t \boldsymbol{U}_t^\top)$ in the regularization can be interpreted as minimizing the *projection F-norm distance* which is defined and denoted for any subspaces S and S' by $\delta_{\mathrm{pF}}(S,S') = \|\boldsymbol{U}\boldsymbol{U}^\top - \boldsymbol{U}'\boldsymbol{U}'^\top\|_{\mathrm{F}}$, where $\|\boldsymbol{M}\|_{\mathrm{F}} = \sqrt{\mathrm{Tr}(\boldsymbol{M}^\top \boldsymbol{M})}$, and $\boldsymbol{U}$ and $\boldsymbol{U}'$ are d-by-k orthogonal basis matrices of S and S' respectively. This follows from the equality $\delta^2_{\mathrm{pF}}(S,S') = 2d - 2\,\mathrm{Tr}(\boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{U}'\boldsymbol{U}'^\top)$. $\delta_{\mathrm{pF}}(S,S')$, and thus the regularization term in Eq. (4.4), are invariant to the choice of $\boldsymbol{U}_s$ and $\boldsymbol{U}_t$. A nice property of the projection F-norm distance is that for subspaces with small geodesic distance, it is asymptotically equivalent to other several important measures including that induced by the intrinsic geometry of the Grassmann manifold (Chevallier et al., 2013; Edelman et al., 1998).

As already mentioned, the multidimensional PCA loss function is invariant under the group action $\boldsymbol{U} \mapsto \boldsymbol{U}\boldsymbol{O}$ for all orthogonal matrices $\boldsymbol{O}$ of size $k \times k$. Hence, optimizing on the space of orthogonal skinny matrices (i.e. the Stiefel manifold) without taking into account

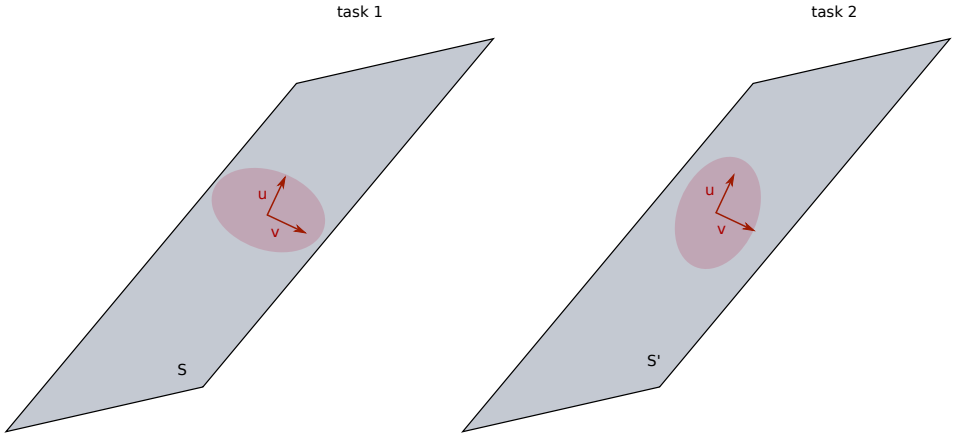


Figure 4.2: Illustration of the invariance of subspaces to the choice of basis. In this 3-dimensional example, the two tasks are generated from the same distribution, but due to sampling, the order of the two main eigenvectors is changed (even though the subspaces are the same). Hence, if we are interested in comparing subspaces, our regularizer should be immune to the choice of bases.

this invariance would be inefficient as the critical points of the cost function are not isolated on the Stiefel manifold. Then, such a property should be taken into account for defining a multi-task regularization. It can be easily shown that this is the case for the proposed regularization of our work since for any orthogonal matrices $\boldsymbol{O}$ and $\boldsymbol{O'}$ of size $k \times k$, we have

$$\mathrm{Tr}(\boldsymbol{U}\underbrace{\boldsymbol{O}\boldsymbol{O}^\top}_{\boldsymbol{I}_k}\boldsymbol{U}^\top\boldsymbol{U}'\underbrace{\boldsymbol{O}'\boldsymbol{O}'^\top}_{\boldsymbol{I}_k}\boldsymbol{U}'^\top) = \mathrm{Tr}(\boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{U}'\boldsymbol{U}'^\top).$$

It would have been tempting to use a simpler regularization such as the matrix scalar product $\mathrm{Tr}(\boldsymbol{U}^\top\boldsymbol{U}')$. However, this regularizer is not invariant under the group action over the product of Grassmann manifolds and this may have some bad consequences. In cases where the top k eigenvalues of a covariance matrix are close, it can happen that the value of those eigenvalues are different (and hence their order changed) for the estimated covariance. Such a situation in two dimensions is illustrated in Figure 4.2. In this case, the subspaces S and S' are identical and respectively represented by the basis matrices $\boldsymbol{U} = \left[\begin{array}{c|c} \boldsymbol{u} & \boldsymbol{v} \end{array}\right]$ and $\boldsymbol{U}' = \left[\begin{array}{c|c} \boldsymbol{v} & \boldsymbol{u} \end{array}\right]$, with $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$ such that $\boldsymbol{u}^\top\boldsymbol{u} = 1$, $\boldsymbol{v}^\top\boldsymbol{v} = 1$ and $\boldsymbol{u}^\top\boldsymbol{v} = 0$. Then, it naturally follows that: $\mathrm{Tr}(\boldsymbol{U}^\top\boldsymbol{U}') = 0 \quad \text{and} \quad \mathrm{Tr}(\boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{U}'\boldsymbol{U}'^\top) = 2$.

When dealing with covariance matrices estimated from few samples, it can happen that the order of the principal eigenvectors is changed compared to the principal eigenvectors of the population covariance. Compared to the naive regularization, our regularization is robust to such a practical problem.

4.2.4 Optimization on Product of Grassmann Manifolds

The Grassmann manifold is a powerful mathematical tool for modeling low-rank transformations, and as noted in Edelman et al. (1998), it is usually involved for solving eigenvalue

problems. As it directly models fixed dimensionality subspaces, it is independent of the bases chosen to represent the subspaces. Hence, as described in Absil et al. (2009, Section 3.4.4), a Grassmann manifold is a quotient manifold and the group structure enables us to encode the invariance properties. In few words, if two representations have the same span, they are said to be equivalent. For a comprehensive tour on this topic, the reader is suggested to refer to Absil et al. (2009); Edelman et al. (1998).

In this work, instead of modeling our dimensionality reduction problem as an optimization problem under a set of orthonormality constraints, we write it as an unconstrained optimization on Grassmann manifolds. Hence, our approach consists in finding several lower-dimensional subspaces by optimizing several transformations (parameterized by $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_T$) that maximize the variance on each dataset meanwhile being similar. As each parameter $\boldsymbol{U}_t$ lies in a Grassmann manifold $\mathrm{Gr}(d, k)$ (Absil et al., 2009; Edelman et al., 1998), we solve the optimization problem on the product of these manifolds.

In Ma et al. (2001), the authors proved that the geodesics in the product manifold are the products of the geodesics in the factor manifolds. This helpful property enables us to compute the gradients on each of the factor manifolds separately and hence to apply easily the machinery of the field of optimization on Riemannian manifolds.

Optimization on Riemannian matrix manifolds is a mature field and by now most of the classical optimization algorithms have been extended to this setting (Absil et al., 2009). In this setting, descent directions are not straight lines but rather curves on the manifold. For a function $f(\boldsymbol{U})$, applying a Riemannian gradient descent can be expressed by the following steps:

1. At any iteration, at the point $\boldsymbol{U}$, transform a Euclidean gradient $D_{\boldsymbol{U}} f$ into a Riemannian gradient $\nabla_{\boldsymbol{U}} f$. In our case, $\nabla_{\boldsymbol{U}} f = D_{\boldsymbol{U}} f - \boldsymbol{U}\boldsymbol{U}^\top D_{\boldsymbol{U}} f$ (Absil et al., 2009).

2. Perform a line search along geodesics at $\boldsymbol{U}$ in the direction $H = \nabla_{\boldsymbol{U}} f$. In our case, on the geodesic going from a point U in direction H (with a step-size t), a new iterate is obtained as $\boldsymbol{U}(t) = \boldsymbol{U}\boldsymbol{V}\cos(\boldsymbol{\Sigma} t)\boldsymbol{V}^\top + \boldsymbol{W}\sin(\boldsymbol{\Sigma} t)\boldsymbol{V}^\top$, where $\boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is the compact singular value decomposition of H.

Our cost function being defined in Eq. (4.4), its Euclidean gradient (w.r.t. a given task t) can be written as:

$$D_{\boldsymbol{U}_t} J = \widehat{\boldsymbol{C}}_t \boldsymbol{U}_t + \lambda \sum_{s \in [T] \setminus \{t\}} \boldsymbol{U}_s \boldsymbol{U}_s^\top \boldsymbol{U}_t. \tag{4.5}$$

In practice, we employ a more sophisticated second-order algorithm called *Riemannian trust-region method* described in Absil et al. (2009) and efficiently implemented in Boumal et al. (2014).

4.3 Experiments

In this section, we present numerical experiments on synthetic and real-life data in order to study the effect of the proposed regularization. We run the proposed method with various regularization parameter values and in various conditions to understand how the performance of the proposed method shifts as the regularization level changes. In this experiment, we compare the performance of the proposed method to the performances of independently applying the PCA to each task (noted as I-PCA and corresponding to the case of $\lambda = 0$) and applying a single PCA over all the datasets (noted as C-PCA and corresponding to the case of $\lambda = \infty$)[*3].

In our *scarce setup*, every task has only scarce data, and the goal is to estimate the optimal subspaces accurately for all the tasks.

4.3.1 Setup

In the scarce setup, we estimate the optimal subspaces with the proposed method using a small number of training samples under several configurations, and then evaluate the quality of the obtained estimates using a large number of test samples. The specific numbers of training and test samples differ from dataset to dataset. We will provide the information in Section 4.3.2.

In the evaluation phase, we measure how much ratio of the variance is preserved when the test sample points are projected onto the estimated subspaces. We refer to this ratio as the *retained variance ratio (RVR)*. We calculate the RVR for every subject t by

$$r_t = \frac{\mathrm{Tr}(\widehat{\boldsymbol{U}}_t^\top \widehat{\boldsymbol{C}}_t' \widehat{\boldsymbol{U}}_t)}{\mathrm{Tr}(\widehat{\boldsymbol{C}}_t')}, \tag{4.6}$$

where $\widehat{\boldsymbol{U}}_t$ denotes an arbitrary basis matrix of the estimated subspace, $\widehat{\boldsymbol{C}}_t'$ is the sample covariance matrix calculated using test samples. Then, we average $r_1, \ldots, r_T$ to obtain the overall score: $r = \frac{1}{T}\sum_{t=1}^{T} r_t$.

In regularization parameter selection by cross-validation, we also use this score but calculated with hold-out samples in place of the test samples.

For statistically reliable evaluation, we run several trials of this experiment with different data realizations[*4]. The specific numbers of trials will be provided in Section 4.3.2.

4.3.2 Data

We tested the method on the following synthetic data and BCI data.

[*3] Note that the method of Wang et al. (2016) being fundamentally a rank-1 method, and as it relies on several hyper-parameters (the number of dictionary atoms and the sparsity level). For these reasons, we decided not to include it in our comparisons.

[*4] By "data realization", we indicate data instances generated with a pseudo random generator in the case of the synthetic dataset, and re-sampled data points from the dataset in the case of the BCI data.

Synthetic Data Sample points for each task t are drawn from the 6-dimensional Gaussian distribution with mean zero and covariance matrix $\boldsymbol{C}_t$ generated in the following way. First, we prepare the 'core' covariance matrix $\boldsymbol{C}_0$ as $\boldsymbol{C}_0 = \boldsymbol{O}_0 \boldsymbol{\Sigma}_0 \boldsymbol{O}_0^\top$, where $\boldsymbol{O}_0 \in \mathbb{R}^{d \times d}$ is a random orthogonal matrix, and $\boldsymbol{\Sigma}_0$ is the diagonal matrix whose diagonal elements are $1, 1, 2, 2, 3, 3$. Second, for each task t, we slightly 'tilt' $\boldsymbol{C}_0$ in order to obtain the task specific covariance $\boldsymbol{C}_t$: $\boldsymbol{C}_t = \boldsymbol{O}_t \boldsymbol{C}_0 \boldsymbol{O}_t^\top$, where $\boldsymbol{O}_t \in \mathbb{R}^{d \times d}$ is an orthogonal matrix nearly equal to $\boldsymbol{I}_d$. We generate $\boldsymbol{O}_t$ as the projected point of $\boldsymbol{I}_d + \boldsymbol{N}$ onto the space of orthogonal matrices[*5], where $\boldsymbol{N}$ is the noise matrix whose elements are i.i.d. samples drawn from the Gaussian distribution with mean zero and variance 0.3.

We conduct the experiment for $k = 1, \ldots, 5$ on this dataset. In the scarce setup, the training sample size is 10 for every task. The test sample size is 10000.

BCI Data This dataset consists of *electroencephalogram (EEG)* signals made available in the context of the *BCI competition IV dataset IIa* (Naeem et al., 2006). This data set is made of EEG signals (recorded from 22 electrodes) from 9 subjects who performed left-hand, right-hand, foot and tongue imaginary movements. As in Yger et al. (2015), we focus on the hand signals (72 trials for each class). This classical paradigm of motor imagination is used for building BCI so that a patient can send commands to a computer by performing imaginary actions.

Then the challenge remains for the computer to accurately detect the correct signal pattern. Nowadays, covariance matrices of EEG signals are commonly used as features for training BCIs (Yger, 2013). In this area, it is time consuming to gather data for a given subject but the data of several subjects are available.

Hence, in this context, our first task will be to investigate the performance of the proposed method in principal subspace extraction of the signals of all the subjects given only the covariance matrix of 1 epoch per subject (Section 4.3.3.1).

Furthermore, we tackle the second task, regularization parameter selection by 2-fold cross-validation, under the setup where two covariance matrices are available (Section 4.3.3.2).

We conduct these experiments for $k = 1, 4, 7, 10$. We sequentially pick one/two epoch(s) (as described above) for each task for subspace estimation, and then the rest of the epochs are used for evaluation of the estimates. We run 72 iterations in the experiment in Section 4.3.3.1 and 36 iterations in the experiment in Section. 4.3.3.2.

4.3.3 Results

We show the results of the experiments below.

4.3.3.1 Performance Transition under Regularization-Level Shift

First, we investigate the performance transition of the proposed method when the regularization level is varied.

[*5] The projection on the space of orthogonal matrices is defined by $\boldsymbol{X} \mapsto \mathrm{argmin}_{\boldsymbol{O} \in \mathbb{R}^{d \times d} : \boldsymbol{O}\boldsymbol{O}^\top = \boldsymbol{I}_d} \|\boldsymbol{X} - \boldsymbol{O}\|_{\mathrm{F}}$.

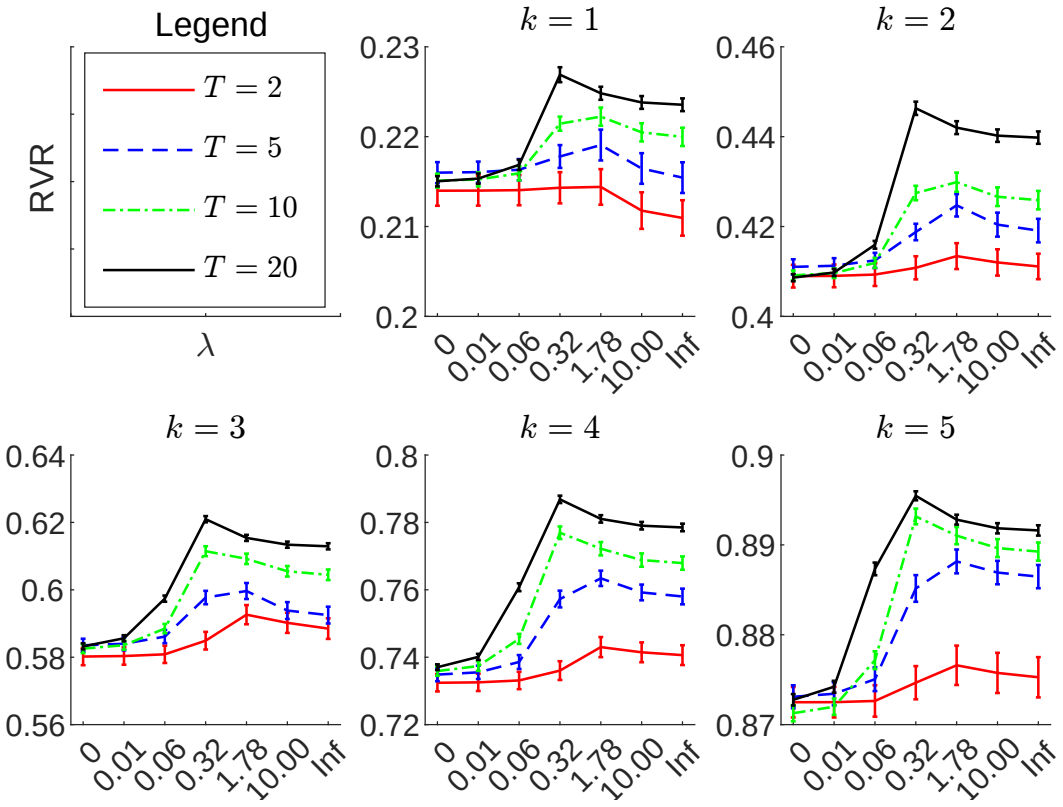


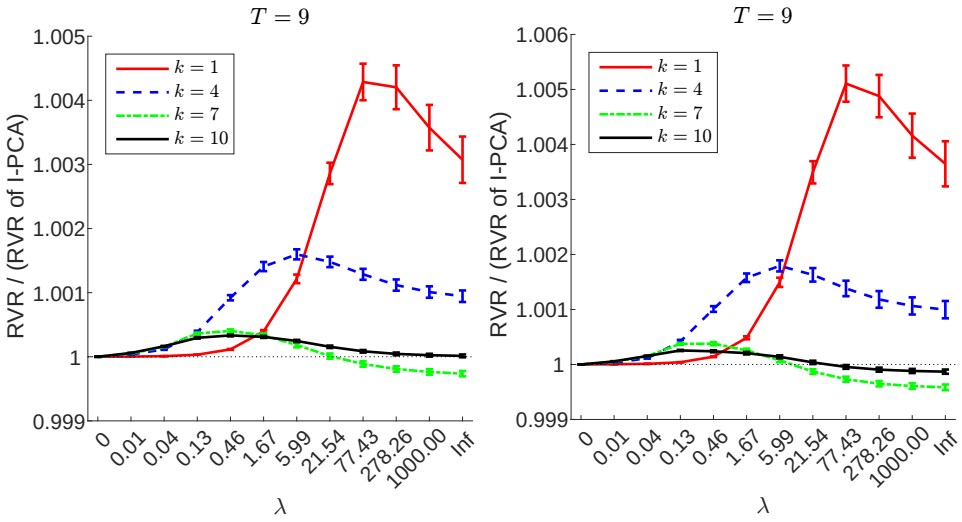
Figure 4.3: The transition of the RVR score over the level of regularization on synthetic data. Each plot corresponds to a different dimensionality k, and each curve corresponds to a different number of tasks T. 'Inf' denotes infinity. The error bars show the mean scores and their standard errors over 100 trials of the experiment.

The results on the synthetic data in the scarce setup are summarized in Figure 4.3. Figure 4.3 shows that the best λ value is somewhere in the middle between 0 and Inf (which denotes infinity) for all of $k = 1, \ldots, 5$, meaning that the proposed method with an appropriate λ value outperforms I-PCA and C-PCA. We can also see the tendency that the performance improves more when we have more tasks.

The results on the BCI data in the scarce setup are shown in Figure 4.4. Similarly to the case of the synthetic data, the performance was improved for all k with appropriate λ values.

The BCI data have most of their variance in a few principal components; the test RVR score for $k = 4$ was more than 96% in all the trials of our experiments, which means that the largest possible RVR gain is less than 4%. Hence, there is less room for improvement for larger k. Nevertheless, the proposed method significantly improved the performance even in such challenging cases.

These results show that the proposed method is useful as long as the regularization level is in a moderate range.



(A) Single Gaussian ($n = 30$) (B) Single Gaussian ($n = 30$)

Figure 4.4: The transition of the RVR score of the proposed method divided by the score of I-PCA over the the level of regularization on BCI data ('Inf' denotes infinity). Each plot corresponds to a different class, and each curve corresponds to a different dimensionality k. The black dotted lines indicate ratio of 1. The error bars show the mean scores and their standard errors over 72 trials.

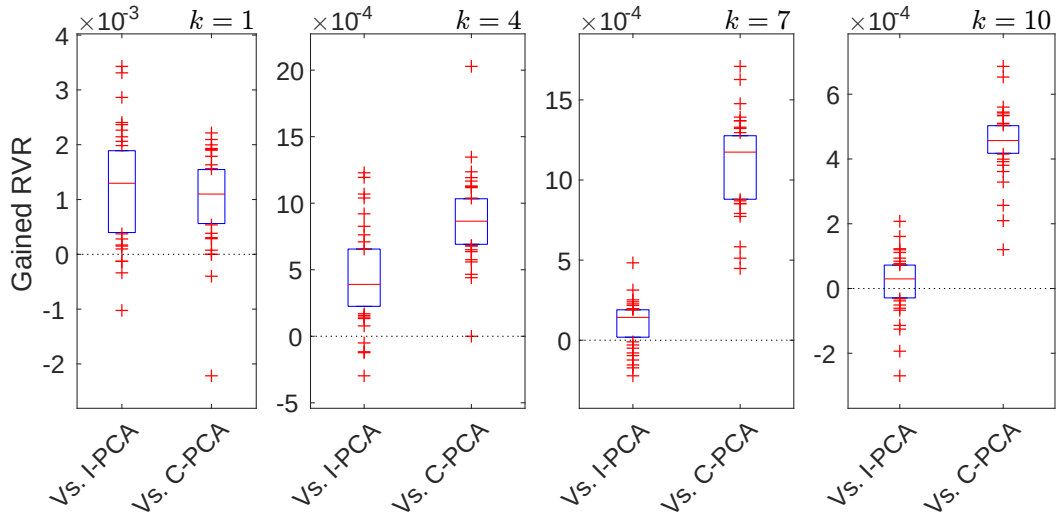
Table 4.1: Averages and standard errors of the RVRs on BCI data. The best and comparable to the best scores by the paired t-test (5% significance level) are shown in bold face.

		CV-MTL	Independent	Common
(Class 1)	$k = 1$	**0.7997(0.0001)**	0.7985(0.0002)	0.7987(0.0001)
	$k = 4$	**0.9670(0.0001)**	0.9666(0.0001)	0.9662(0.0001)
	$k = 7$	**0.9877(0.0000)**	0.9876(0.0000)	0.9866(0.0000)
	$k = 10$	**0.9945(0.0000)**	**0.9945(0.0000)**	0.9941(0.0000)
(Class 2)	$k = 1$	**0.7857(0.0001)**	0.7844(0.0003)	0.7844(0.0001)
	$k = 4$	**0.9655(0.0001)**	0.9651(0.0001)	0.9646(0.0000)
	$k = 7$	**0.9872(0.0000)**	0.9871(0.0000)	0.9859(0.0000)
	$k = 10$	**0.9943(0.0000)**	**0.9943(0.0000)**	0.9938(0.0000)

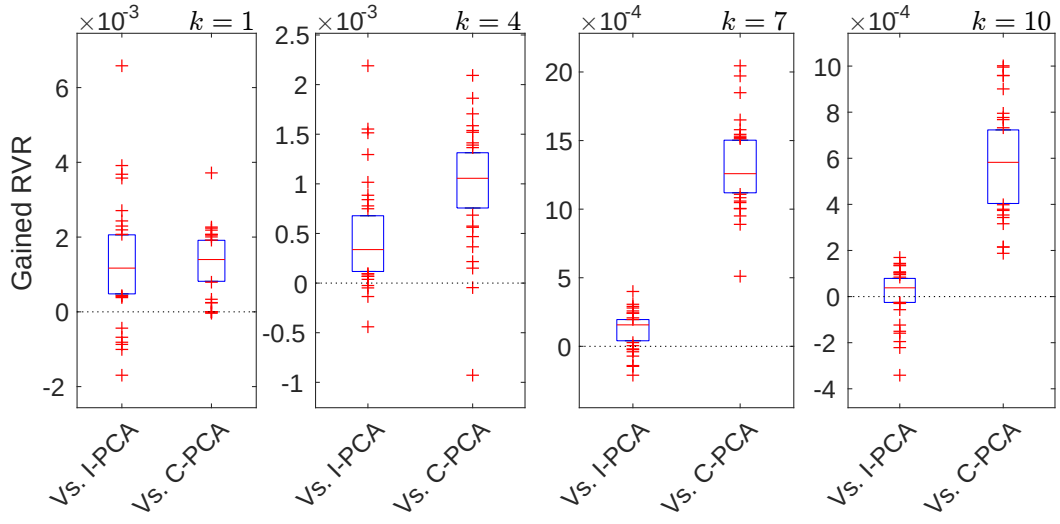
4.3.3.2 Regularization Parameter Selection by Cross-Validation

In Section 4.3.3.1, the experiments in the scarce setup showed that there exists a regularization parameter such that our method achieves better results than those of the baseline methods. In order to select such a parameter, we apply a cross-validation method and provide some experimental results on BCI data. On the BCI data, the proposed method outperformed the other two methods for all of $k = 1, 4, 7, 10$ on average (see Table 4.1). The box plots in Figure 4.5 detail the results, showing that RMT-PCA scored larger RVRs compared to I-PCA and C-PCA in most of the trials in every setting.

From these experiments, it is demonstrated that the proposed method with a regularization parameter automatically selected by cross-validation performs significantly better than I-PCA and C-PCA.



(A) Class 1



(B) Class 2

Figure 4.5: The RVR of the proposed method using a cross-validated regularization parameter substracted by the RVRs of its competitors (I-PCA and C-PCA) on BCI data. The samples between the 25% and the 75% quantiles are summarized as a blue box and the rest are shown as red + symbols in each plot.

4.4 Conclusion

In this chapter, we introduced a novel regularization term for orthogonal skinny matrices. Based on this regularization term, we provided a novel and elegant formulation of the multi-task PCA problem. Using tools from the field of optimization on manifolds, we solved this problem, applied our method to synthetic and real-world data, and demonstrated its usefulness.

We only considered multi-task learning in the scarce setting, but the proposed regularization can be applied to transfer learning and adaptation problems, whose goals are to improve the performance for a single target task utilizing the information from other similar tasks. Real-world examples where multi-task principal component analysis plays important roles include analysis of multi-country government bond returns (Pérignon et al., 2007) and preprocessing for learning biometric verification systems (Delac and Grgic, 2004). The particular usefulness of principal component analysis in face image processing with scarce samples is argued in Jafri and Arabnia (2009).

In future work, we consider several extensions of our method. We may cast our multi-task dimensionality reduction to a supervised setup. Such an approach may be particularly useful for BCI applications.

Other subspace methods such as locality preserving projections (He and Niyogi, 2004), Fisher's discriminant analysis (Fisher, 1936), and canonical correlation analysis (Hotelling, 1936) can be extended to multi-task scenarios using the proposed regularization by replacing the sample covariance $\widehat{C}_t$ in Eq. (4) with appropriate symmetric matrices.

In addition, it would be interesting to use our approach with different criteria in the spirit of Harandi et al. (2014); Horev et al. (2015), leading to a multi-task Riemannian dimensionality reduction.

Chapter 5

Uplift Modeling from Separate Labels

Uplift modeling is aimed at estimating the incremental impact of an action on an individual's behavior, which is useful in various application domains such as targeted marketing (advertisement campaigns) and personalized medicine (medical treatments). Conventional methods of uplift modeling require every instance to be *jointly* equipped with two types of labels: the taken action and its outcome. However, obtaining two labels for each instance at the same time is difficult or expensive in many real-world problems. In this chapter, we propose a novel method of uplift modeling that is applicable to a more practical setting where only one type of labels is available for each instance. We show a mean squared error bound for the proposed estimator and demonstrate its effectiveness through experiments.

5.1 Introduction

In many real-world problems, a central objective is to optimally choose a right action to maximize the profit of interest. For example, in marketing, an advertising campaign is designed to promote people to purchase a product (Renault, 2015). A marketer can choose whether to deliver an advertisement to each individual or not, and the outcome is the number of purchases of the product. Another example is personalized medicine, where a treatment is chosen depending on each patient to maximize the medical effect and minimize the risk of adverse events or harmful side effects (Abrahams and Silver, 2009; Katsanis et al., 2008). In this case, giving or not giving a medical treatment to each individual are the possible actions to choose, and the outcome is the rate of recovery or survival from the disease. Hereafter, we use the word *treatment* for taking an action, following the personalized medicine example.

A/B testing (Kohavi et al., 2009) is a standard method for such tasks, where two groups of people, A and B, are randomly chosen. The outcomes are measured separately from the two groups after treating all the members of Group A but none of Group B. By comparing the outcomes between the two groups by a statistical test, one can examine whether the treatment positively or negatively affected the outcome. However, A/B testing only compares the two extreme options: treating everyone or no one. These two options can be both far

from optimal when the treatment has positive effect on some individuals but negative effect on others.

To overcome the drawback of A/B testing, *uplift modeling* has been investigated recently (Jaskowski and Jaroszewicz, 2012; Radcliffe and Surry, 2011; Rzepakowski and Jaroszewicz, 2012a). *Uplift modeling* is the problem of estimating the *individual uplift*, the incremental profit brought by the treatment conditioned on features of each individual. Uplift modeling enables us to design a refined decision rule for optimally determining whether to treat each individual or not, depending on his/her features. Such a treatment rule allows us to only target those who positively respond to the treatment and avoid treating negative responders.

In the standard uplift modeling setup, there are two types of labels (Jaskowski and Jaroszewicz, 2012; Radcliffe and Surry, 2011; Rzepakowski and Jaroszewicz, 2012a): One is whether the treatment has been given to the individual and the other is its outcome. Existing uplift modeling methods require each individual to be *jointly* given these two labels for analyzing the association between outcomes and the treatment (Jaskowski and Jaroszewicz, 2012; Radcliffe and Surry, 2011; Rzepakowski and Jaroszewicz, 2012a). However, joint labels are expensive or hard (or even impossible) to obtain in many real-world problems. For example, when distributing an advertisement by email, we can easily record to whom the advertisement has been sent. However, for technical or privacy reasons, it is difficult to keep track of those people until we observe the outcomes on whether they buy the product or not. Alternatively, we can easily obtain information about purchasers of the product at the moment when the purchases are actually made. However, we cannot know whether those who are buying the product have been exposed to the advertisement or not. Thus, every individual always has one missing label. We term such samples *separately labeled samples*.

In this work, we consider a more practical uplift modeling setup where no jointly labeled samples are available, but only separately labeled samples are given. Theoretically, we first show that the individual uplift is identifiable when we have two sets of separately labeled samples collected under *different* treatment policies. We then propose a novel method that directly estimates the individual uplift only from separately labeled samples. Finally, we demonstrate the effectiveness of the proposed method through experiments.

5.2 Problem Setting

This work focuses on estimation of the *individual uplift* $u(\boldsymbol{x})$, often called *individual treatment effect (ITE)* in the causal inference literature (Rubin, 2005), defined as $u(\boldsymbol{x}) := \mathbf{E}[Y_1 \mid \boldsymbol{x}] - \mathbf{E}[Y_{-1} \mid \boldsymbol{x}]$, where $\mathbf{E}[\,\cdot \mid \cdot\,]$ denotes the conditional expectation, and $\boldsymbol{x}$ is a $\mathcal{X}$-valued random variable ($\mathcal{X} \subseteq \mathbb{R}^d$) representing features of an individual, and Y_1, Y_{-1} are $\mathcal{Y}$-valued *potential outcome variables* (Rubin, 2005) ($\mathcal{Y} \subseteq \mathbb{R}$) representing outcomes that would be observed if the individual was treated and not treated, respectively. Note that only one of either Y_1 or Y_{-1} can be observed for each individual. We denote the $\{1, -1\}$-valued random variable of the treatment assignment by t, where $t = 1$ means that the individual has been

treated and $t = -1$ not treated. We refer to the population for which we want to evaluate $u(\boldsymbol{x})$ as the *test population*, and denote the density of the test population by $p(Y_1, Y_{-1}, \boldsymbol{x}, t)$.

We assume that t is *unconfounded*[*1] with either of Y_1 and Y_{-1} conditioned on $\boldsymbol{x}$, i.e. $p(Y_1 \mid \boldsymbol{x}, t) = p(Y_1 \mid \boldsymbol{x})$ and $p(Y_{-1} \mid \boldsymbol{x}, t) = p(Y_{-1} \mid \boldsymbol{x})$. Unconfoundedness is an assumption commonly made in observational studies (Gutierrez and Gérardy, 2017; Shalit et al., 2017). For notational convenience, we denote by $y := Y_t$ the outcome of the treatment assignment t. Furthermore, we refer to any conditional density of t given $\boldsymbol{x}$ as a *treatment policy*.

In addition to the test population, we suppose that there are two *training populations* $k = 1, 2$, whose joint probability density $p_k(Y_1, Y_{-1}, \boldsymbol{x}, t)$ satisfy

$$p_k(Y_{t_0} = y_0 \mid \boldsymbol{x} = \boldsymbol{x}_0) = p(Y_{t_0} = y_0 \mid \boldsymbol{x} = \boldsymbol{x}_0) \quad (\text{for } k = 1, 2), \tag{5.1}$$

$$p_1(t = t_0 \mid \boldsymbol{x} = \boldsymbol{x}_0) \neq p_2(t = t_0 \mid \boldsymbol{x} = \boldsymbol{x}_0), \tag{5.2}$$

for all possible realizations $\boldsymbol{x}_0 \in \mathcal{X}$, $t_0 \in \{-1, 1\}$, and $y_0 \in \mathcal{Y}$. Intuitively, Eq. (5.1) means that potential outcomes depend on $\boldsymbol{x}$ in the same way as those in the test population, and Eq. (5.2) states that those two policies give a treatment with different probabilities for every $\boldsymbol{x} = \boldsymbol{x}_0$.

We suppose that the following four training data sets, which we call *separately labeled samples*, are given:

$$\{(\boldsymbol{x}_i^{(k)}, y_i^{(k)})\}_{i=1}^{n_k} \overset{\text{i.i.d.}}{\sim} p_k(\boldsymbol{x}, y), \quad \{(\widetilde{\boldsymbol{x}}_i^{(k)}, t_i^{(k)})\}_{i=1}^{\widetilde{n}_k} \overset{\text{i.i.d.}}{\sim} p_k(\boldsymbol{x}, t) \quad (\text{for } k = 1, 2),$$

where n_k and $\widetilde{n}_k$, $k = 1, 2$, are positive integers. Under Assumptions (5.1), (5.2), and the unconfoundedness, we have $p_k(Y_t \mid \boldsymbol{x}, t = t_0) = p(Y_{t_0} \mid \boldsymbol{x}, t = t_0) = p(Y_{t_0} \mid \boldsymbol{x})$ for $t_0 \in \{-1, 1\}$ and $k \in \{1, 2\}$. Note that we can safely denote $p(y \mid \boldsymbol{x}, t) := p_k(y \mid \boldsymbol{x}, t)$. Moreover, we have $\mathbf{E}[Y_{t_0} \mid \boldsymbol{x}] = \mathbf{E}[y \mid \boldsymbol{x}, t = t_0]$ for $t_0 = 1, -1$, and thus our goal boils down to the estimation of

$$u(\boldsymbol{x}) = \mathbf{E}[y \mid \boldsymbol{x}, t = 1] - \mathbf{E}[y \mid \boldsymbol{x}, t = -1] \tag{5.3}$$

from the separately labeled samples, where the conditional expectation is taken over $p(y \mid \boldsymbol{x}, t)$.

Estimation of the individual uplift is important for the following reasons.

It enables the estimation of the average uplift. The *average uplift* $U(\pi)$ of the treatment policy $\pi(t \mid \boldsymbol{x})$ is the average outcome of π, subtracted by that of the policy π_-, which constantly assigns the treatment as $t = -1$, i.e., $\pi_-(t = \tau \mid \boldsymbol{x}) := 1[\tau = -1]$, where

[*1] This condition is also referred to as *exchangeability*.

$1[\cdot]$ denotes the indicator function:

$$U(\pi) := \iint \sum_{t=-1,1} yp(y \mid \boldsymbol{x}, t)\pi(t \mid \boldsymbol{x})p(\boldsymbol{x})\mathrm{d}y\mathrm{d}\boldsymbol{x} - \iint \sum_{t=-1,1} yp(y \mid \boldsymbol{x}, t)\pi_{-}(t \mid \boldsymbol{x})p(\boldsymbol{x})\mathrm{d}y\mathrm{d}\boldsymbol{x}$$
$$= \int u(\boldsymbol{x})\pi(t = 1 \mid \boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \tag{5.4}$$

This quantity can be estimated from samples of $\boldsymbol{x}$ once we obtain an estimate of $u(\boldsymbol{x})$.

It provides the optimal treatment policy. The treatment policy given by $\pi(t = 1 \mid \boldsymbol{x}) = 1[0 \leq u(\boldsymbol{x})]$ is the optimal treatment that maximizes the average uplift $U(\pi)$ and equivalently the average outcome $\iint \sum_{t=-1,1} yp(y \mid \boldsymbol{x}, t)\pi(t \mid \boldsymbol{x})p(\boldsymbol{x})\mathrm{d}y\mathrm{d}\boldsymbol{x}$ (see Eq. (5.4)) (Rzepakowski and Jaroszewicz, 2012a).

It is the optimal ranking scoring function. From a practical viewpoint, it may be useful to prioritize individuals to be treated according to some ranking scores especially when the treatment is costly and only a limited number of individuals can be treated due to some budget constraint. In fact, $u(\boldsymbol{x})$ serves as the optimal ranking scores for this purpose (Tufféry, 2011). More specifically, we define a family of treatment policies $\{\pi_{f,\alpha}\}_{\alpha\in\mathbb{R}}$ *associated with scoring function* f by $\pi_{f,\alpha}(t = 1 \mid \boldsymbol{x}) = 1[\alpha \leq f(\boldsymbol{x})]$. Then, under some technical condition, $f = u$ maximizes the *area under the uplift curve (AUUC)* defined as

$$\begin{aligned}\mathrm{AUUC}(f) &:= \int_0^1 U(\pi_{f,\alpha})\mathrm{d}C_\alpha \\ &= \int_0^1 \int u(\boldsymbol{x})1[\alpha \leq f(\boldsymbol{x})]p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\mathrm{d}C_\alpha \\ &= \mathbf{E}[1[f(\boldsymbol{x}) \leq f(\boldsymbol{x}')]u(\boldsymbol{x}')],\end{aligned}$$

where $C_\alpha := \Pr[f(\boldsymbol{x}) < \alpha]$, $\boldsymbol{x}, \boldsymbol{x}' \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x})$, and $\mathbf{E}$ denotes the expectation with respect to these variables. AUUC is a standard performance measure for uplift modeling methods (Jaskowski and Jaroszewicz, 2012; Radcliffe, 2007; Radcliffe and Surry, 2011; Rzepakowski and Jaroszewicz, 2012a). For more details, see Appendix A.2.

Remark on the problem setting: Uplift modeling is often referred to as individual treatment effect estimation or heterogeneous treatment effect estimation and has been extensively studied especially in the causal inference literature (Gutierrez and Gérardy, 2017; Hill, 2011; Imai and Ratkovic, 2013; Johansson et al., 2016; Künzel et al., 2017; Pearl, 2009; Rubin, 2005; Wager and Athey, 2015). In particular, recent research has investigated the problem under the setting of *observational studies*, inference using data obtained from *uncontrolled experiments* because of its practical importance (Shalit et al., 2017). Here, experiments are said to be uncontrolled when some of treatment variables are not controlled to have designed values.

Given that treatment policies are unknown, our problem setting is also of observational studies but poses an additional challenge that stems from missing labels. What makes our problem feasible is that we have two kinds of data sets following different treatment policies.

It is also important to note that our setting generalizes the standard setting for observational studies since the former is reduced to the latter when one of the treatment policies always assigns individuals to the treatment group, and the other to the control group.

Our problem is also closely related to individual treatment effect estimation via instrumental variables (Athey et al., 2016; Hartford et al., 2017; Imbens, 2014; Lewis and Syrgkanis, 2018).[*2]

5.3 Naive Estimators

A naive approach is first estimating the conditional density $p_k(y \mid \boldsymbol{x})$ and $p_k(t \mid \boldsymbol{x})$ from training samples by some conditional density estimator (Bishop, 2006b; Sugiyama et al., 2010), and then solving the following linear system for $p(y \mid \boldsymbol{x}, t = 1)$ and $p(y \mid \boldsymbol{x}, t = -1)$:

$$\underbrace{p_k(y \mid \boldsymbol{x})}_{\text{Estimated from } \{(\boldsymbol{x}_i^{(k)}, y_i^{(k)})\}_{i=1}^{n}} = \sum_{t=-1,1} p(y \mid \boldsymbol{x}, t) \underbrace{p_k(t \mid \boldsymbol{x})}_{\text{Estimated from } \{(\widetilde{\boldsymbol{x}}_i^{(k)}, t_i^{(k)})\}_{i=1}^{\widetilde{n}}} \quad (\text{for } k = 1, 2). \tag{5.5}$$

After that, the conditional expectations of y over $p(y \mid \boldsymbol{x}, t = 1)$ and $p(y \mid \boldsymbol{x}, t = -1)$ are calculated by numerical integration, and finally their difference is calculated to obtain another estimate of $u(\boldsymbol{x})$.

However, this may not yield a good estimate due to the difficulty of conditional density estimation and the instability of numerical integration. This issue may be alleviated by working on the following linear system implied by Eq. (5.5) instead: $\mathbf{E}_k[y \mid \boldsymbol{x}] = \sum_{t=-1,1} \mathbf{E}[y \mid \boldsymbol{x}, t] p_k(t \mid \boldsymbol{x})$, $k = 1, 2$, where $\mathbf{E}_k[y \mid \boldsymbol{x}]$ and $p_k(t \mid \boldsymbol{x})$ can be estimated from our samples. Solving this new system for $\mathbf{E}[y \mid \boldsymbol{x}, t = 1]$ and $\mathbf{E}[y \mid \boldsymbol{x}, t = -1]$ and taking their difference gives an estimate of $u(\boldsymbol{x})$. A method called *two-stage least-squares* for instrumental variable regression takes such an approach (Imbens, 2014).

The second approach of estimation $E_k[y|x]$ and $p_k(t|x)$ avoids both conditional density estimation and numerical integration, but it still involves post processing of solving the linear system and subtraction, being a potential cause of performance deterioration.

5.4 Proposed Method

In this section, we develop a method that can overcome the aforementioned problems by directly estimating the individual uplift.

[*2] Among the related papers mentioned above, the most relevant one is Lewis and Syrgkanis (2018), which is concurrent work with ours.

5.4.1 Direct Least-Square Estimation of the Individual Uplift

First, we will show an important lemma that directly relates the marginal distributions of separately labeled samples to the individual uplift $u(\boldsymbol{x})$.

Lemma 5.4.1. *For every $\boldsymbol{x}$ such that $p_1(\boldsymbol{x}) \neq p_2(\boldsymbol{x})$, $u(\boldsymbol{x})$ can be expressed as*

$$u(\boldsymbol{x}) = 2 \times \frac{\mathbf{E}_{y \sim p_1(y|\boldsymbol{x})}[y] - \mathbf{E}_{y \sim p_2(y|\boldsymbol{x})}[y]}{\mathbf{E}_{t \sim p_1(t|\boldsymbol{x})}[t] - \mathbf{E}_{t \sim p_2(t|\boldsymbol{x})}[t]}. \tag{5.6}$$

For a proof, refer to Appendix A.3.

Using Eq. (5.6), we can re-interpret the naive methods described in Section 5.3 as estimating the conditional expectations on the right-hand side by separately performing regression on $\{(\boldsymbol{x}_i^{(1)}, y_i^{(1)})\}_{i=1}^{n_1}$, $\{(\boldsymbol{x}_i^{(2)}, y_i^{(2)})\}_{i=1}^{n_2}$, $\{(\widetilde{\boldsymbol{x}}_i^{(1)}, t_i^{(1)})\}_{i=1}^{\widetilde{n}_1}$, and $\{(\widetilde{\boldsymbol{x}}_i^{(2)}, t_i^{(2)})\}_{i=1}^{\widetilde{n}_2}$. This approach may result in unreliable performance when the denominator is close to zero, i.e., $p_1(t \mid \boldsymbol{x}) \simeq p_2(t \mid \boldsymbol{x})$.

Lemma 5.4.1 can be simplified by introducing auxiliary variables z and w, which are $\mathcal{Z}$-valued and $\{-1, 1\}$-valued random variables whose conditional probability density and mass are defined by

$$\begin{aligned} p(z = z_0 \mid \boldsymbol{x}) &= \tfrac{1}{2} p_1(y = z_0 \mid \boldsymbol{x}) + \tfrac{1}{2} p_2(y = -z_0 \mid \boldsymbol{x}), \\ p(w = w_0 \mid \boldsymbol{x}) &= \tfrac{1}{2} p_1(t = w_0 \mid \boldsymbol{x}) + \tfrac{1}{2} p_2(t = -w_0 \mid \boldsymbol{x}), \end{aligned}$$

for any $z_0 \in \mathcal{Z}$ and any $w_0 \in \{-1, 1\}$, where $\mathcal{Z} := \{s_0 y_0 \mid y_0 \in \mathcal{Y}, s_0 \in \{1, -1\}\}$.

Lemma 5.4.2. *For every $\boldsymbol{x}$ such that $p_1(\boldsymbol{x}) \neq p_2(\boldsymbol{x})$, $u(\boldsymbol{x})$ can be expressed as*

$$u(\boldsymbol{x}) = 2 \times \frac{\mathbf{E}[z \mid \boldsymbol{x}]}{\mathbf{E}[w \mid \boldsymbol{x}]},$$

where $\mathbf{E}[z \mid \boldsymbol{x}]$ and $\mathbf{E}[w \mid \boldsymbol{x}]$ are the conditional expectations of z given $\boldsymbol{x}$ over $p(z \mid \boldsymbol{x})$ and w given $\boldsymbol{x}$ over $p(w \mid \boldsymbol{x})$, respectively.

A proof can be found in Appendix A.4.

Let $w_i^{(k)} := (-1)^{k-1} t_i^{(k)}$ and $z_i^{(k)} := (-1)^{k-1} y_i^{(k)}$. Assuming that $p_1(\boldsymbol{x}) = p_2(\boldsymbol{x}) =: p(\boldsymbol{x})$, $n_1 = n_2$, and $\widetilde{n}_1 = \widetilde{n}_2$ for simplicity, $\{(\widetilde{\boldsymbol{x}}_i, w_i)\}_{i=1}^{n} := \{(\widetilde{\boldsymbol{x}}_i^{(k)}, w_i^{(k)})\}_{k=1,2;\ i=1,\ldots,\widetilde{n}_k}$ and $\{(\boldsymbol{x}_i, z_i)\}_{i=1}^{n} := \{(\boldsymbol{x}_i^{(k)}, z_i^{(k)})\}_{k=1,2;\ i=1,\ldots,n_k}$ can be seen as samples drawn from $p(\boldsymbol{x}, z) := p(z \mid \boldsymbol{x}) p(\boldsymbol{x})$ and $p(\boldsymbol{x}, w) := p(w \mid \boldsymbol{x}) p(\boldsymbol{x})$, respectively, where $n = n_1 + n_2$ and $\widetilde{n} = \widetilde{n}_1 + \widetilde{n}_2$. The more general cases where $p_1(\boldsymbol{x}) \neq p_2(\boldsymbol{x})$, $n_1 \neq n_2$, or $\widetilde{n}_1 \neq \widetilde{n}_2$ are discussed in Appendix A.9.

Theorem 5.4.1. *Assume that $\mu_w, \mu_z \in L^2(p)$ and $\mu_w(\boldsymbol{x}) \neq 0$ for every $\boldsymbol{x}$ such that $p(\boldsymbol{x}) > 0$, where $L^2(p) := \{f : \mathcal{X} \to \mathbb{R} \mid \mathbf{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})}[f(\boldsymbol{x})^2] < \infty\}$. The individual uplift $u(\boldsymbol{x})$ equals the solution to the following least-squares problem:*

$$u(\boldsymbol{x}) = \operatorname*{argmin}_{f \in L^2(p)} \mathbf{E}[(\mu_w(\boldsymbol{x}) f(\boldsymbol{x}) - 2\mu_z(\boldsymbol{x}))^2], \tag{5.7}$$

where $\mathbf{E}$ *denotes the expectation over* $p(\boldsymbol{x})$, $\mu_w(\boldsymbol{x}) := \mathbf{E}[w \mid \boldsymbol{x}]$, *and* $\mu_z(\boldsymbol{x}) := \mathbf{E}[z \mid \boldsymbol{x}]$.

Theorem 5.4.1 follows from Lemma 5.4.2. Note that $p_1(\boldsymbol{x}) \neq p_2(\boldsymbol{x})$ in Eq. (5.2) implies $\mu_w(\boldsymbol{x}) \neq 0$.

In what follows, we develop a method that directly estimates $u(\boldsymbol{x})$ by solving Eq. (5.7). A challenge here is that it is not straightforward to evaluate the objective functional since it involves unknown functions, μ_w and μ_z.

5.4.2 Disentanglement of z and w

Our idea is to transform the objective functional in Eq. (5.7) into another form in which $\mu_w(\boldsymbol{x})$ and $\mu_z(\boldsymbol{x})$ appear separately and linearly inside the expectation operator so that we can approximate them using our separately labeled samples.

For any function $g \in L^2(p)$ and any $\boldsymbol{x} \in \mathcal{X}$, expanding the left-hand side of the inequality $\mathbf{E}[(\mu_w(\boldsymbol{x})f(\boldsymbol{x}) - 2\mu_z(\boldsymbol{x}) - g(\boldsymbol{x}))^2] \geq 0$, we have

$$\mathbf{E}[(\mu_w(\boldsymbol{x})f(\boldsymbol{x}) - 2\mu_z(\boldsymbol{x}))^2] \geq 2\,\mathbf{E}[(\mu_w(\boldsymbol{x})f(\boldsymbol{x}) - 2\mu_z(\boldsymbol{x}))g(\boldsymbol{x})] - \mathbf{E}[g(\boldsymbol{x})^2] =: J(f,g). \quad (5.8)$$

The equality is attained when $g(\boldsymbol{x}) = \mu_w(\boldsymbol{x})f(\boldsymbol{x}) - \mu_z(\boldsymbol{x})$ for any fixed f. This means that the objective functional of Eq. (5.7) can be calculated by maximizing $J(f,g)$ with respect to g. Hence,

$$u(\boldsymbol{x}) = \operatorname*{argmin}_{f \in L^2(p)} \max_{g \in L^2(p)} J(f,g). \quad (5.9)$$

Furthermore, μ_w and μ_z are separately and linearly included in $J(f,g)$, which makes it possible to write it in terms of z and w as

$$J(f,g) = 2\,\mathbf{E}[wf(\boldsymbol{x})g(\boldsymbol{x})] - 4\,\mathbf{E}[zg(\boldsymbol{x})] - \mathbf{E}[g(\boldsymbol{x})^2]. \quad (5.10)$$

Unlike the original objective functional in Eq. (5.7), $J(f,g)$ can be easily estimated using sample averages by

$$\widehat{J}(f,g) = \frac{2}{\widetilde{n}} \sum_{i=1}^{\widetilde{n}} w_i f(\widetilde{\boldsymbol{x}}_i) g(\widetilde{\boldsymbol{x}}_i) - \frac{4}{n} \sum_{i=1}^{n} z_i g(\boldsymbol{x}_i) - \frac{1}{2n} \sum_{i=1}^{n} g(\boldsymbol{x}_i)^2 - \frac{1}{2\widetilde{n}} \sum_{i=1}^{\widetilde{n}} g(\widetilde{\boldsymbol{x}}_i)^2. \quad (5.11)$$

In practice, we solve the following regularized empirical optimization problem:

$$\min_{f \in F} \max_{g \in G} \widehat{J}(f,g) + \Omega(f,g), \quad (5.12)$$

where F, G are models for f, g respectively, and $\Omega(f,g)$ is some regularizer. An advantage of the proposed framework is that it is model-independent, and any models can be trained by optimizing the above objective.

The function g can be interpreted as a *critic* of f as follows. Minimizing Eq. (5.10) with respect to f is equivalent to minimizing $J(f,g) = \mathbf{E}[g(\boldsymbol{x})\{\mu_w(\boldsymbol{x})f(\boldsymbol{x}) - 2\mu_z(\boldsymbol{x})\}]$. $g(\boldsymbol{x})$

serves as a good critic of $f(\boldsymbol{x})$ when it makes the cost $g(\boldsymbol{x})\{\mu_w(\boldsymbol{x})f(\boldsymbol{x}) - 2\mu_z(\boldsymbol{x})\}$ larger for $\boldsymbol{x}$ at which f makes a larger error $|\mu_w(\boldsymbol{x})f(\boldsymbol{x}) - 2\mu_z(\boldsymbol{x})|$. In particular, g maximizes the objective above when $g(\boldsymbol{x}) = \mu_w(\boldsymbol{x})f(\boldsymbol{x}) - 2\mu_z(\boldsymbol{x})$ for any f, and the maximum coincides with the least-squares objective in Eq. (5.7).

Suppose that F and G are linear-in-parameter models: $F = \{f_{\boldsymbol{\alpha}} : \boldsymbol{x} \mapsto \boldsymbol{\alpha}^\top \boldsymbol{\phi}(\boldsymbol{x}) \mid \boldsymbol{\alpha} \in \mathbb{R}^{b_f}\}$ and $G = \{g_{\boldsymbol{\beta}} : \boldsymbol{x} \mapsto \boldsymbol{\beta}^\top \boldsymbol{\psi}(\boldsymbol{x}) \mid \boldsymbol{\beta} \in \mathbb{R}^{b_g}\}$, where $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are b_{f}-dimensional and b_{g}-dimensional vectors of basis functions in $L^2(p)$. Then, $\widehat{J}(f_{\boldsymbol{\alpha}}, g_{\boldsymbol{\beta}}) = 2\boldsymbol{\alpha}^\top \boldsymbol{A}\boldsymbol{\beta} - 4\boldsymbol{b}^\top \boldsymbol{\beta} - \boldsymbol{\beta}^\top \boldsymbol{C}\boldsymbol{\beta}$, where

$$\boldsymbol{A} := \frac{1}{\widetilde{n}} \sum_{i=1}^{\widetilde{n}} w_i \boldsymbol{\phi}(\widetilde{\boldsymbol{x}}_i)\boldsymbol{\psi}(\widetilde{\boldsymbol{x}}_i)^\top, \quad \boldsymbol{b} := \frac{1}{n} \sum_{i=1}^{n} z_i \boldsymbol{\psi}(\boldsymbol{x}_i),$$
$$\boldsymbol{C} := \frac{1}{2n} \sum_{i=1}^{n} \boldsymbol{\psi}(\boldsymbol{x}_i)\boldsymbol{\psi}(\boldsymbol{x}_i)^\top + \frac{1}{2\widetilde{n}} \sum_{i=1}^{\widetilde{n}} \boldsymbol{\psi}(\widetilde{\boldsymbol{x}}_i)\boldsymbol{\psi}(\widetilde{\boldsymbol{x}}_i)^\top.$$

Using ℓ_2-regularizers, $\Omega(f, g) = \lambda_{\mathrm{f}} \boldsymbol{\alpha}^\top \boldsymbol{\alpha} - \lambda_{\mathrm{g}} \boldsymbol{\beta}^\top \boldsymbol{\beta}$ with some positive constants λ_{f} and λ_{g}, the solution to the inner maximization problem can be obtained in the following analytical form:

$$\widehat{\boldsymbol{\beta}}_{\boldsymbol{\alpha}} := \operatorname*{argmax}_{\boldsymbol{\beta}} \widehat{J}(f_{\boldsymbol{\alpha}}, g_{\boldsymbol{\beta}}) = \widetilde{\boldsymbol{C}}^{-1}(\boldsymbol{A}^\top \boldsymbol{\alpha} - 2\boldsymbol{b}),$$

where $\widetilde{\boldsymbol{C}} = \boldsymbol{C} + \lambda_{\mathrm{g}} \boldsymbol{I}_{b_{\mathrm{g}}}$ and $\boldsymbol{I}_{b_{\mathrm{g}}}$ is the b_{g}-by-b_{g} identity matrix. Then, we can obtain the solution to Eq. (5.12) analytically as

$$\widehat{\boldsymbol{\alpha}} := \operatorname*{argmin}_{\boldsymbol{\alpha}} \widehat{J}(f_{\boldsymbol{\alpha}}, g_{\widehat{\boldsymbol{\beta}}_{\boldsymbol{\alpha}}}) = 2(\boldsymbol{A}\widetilde{\boldsymbol{C}}^{-1}\boldsymbol{A}^\top + \lambda_{\mathrm{f}} \boldsymbol{I}_{b_{\mathrm{g}}})^{-1} \boldsymbol{A}\widetilde{\boldsymbol{C}}^{-1}\boldsymbol{b}.$$

Finally, from Eq. (5.7), our estimate of $u(\boldsymbol{x})$ is given as $\widehat{\boldsymbol{\alpha}}^\top \boldsymbol{\phi}(\boldsymbol{x})$.

Remark on model selection: Model selection for F and G is not straightforward since the test performance measure cannot be directly evaluated with (held out) training data of our problem. Instead, we may evaluate the value of $J(\widehat{f}, \widehat{g})$, where $(\widehat{f}, \widehat{g}) \in F \times G$ is the optimal solution pair to $\min_{f \in F} \max_{g \in G} \widehat{J}(f, g)$. However, it is still nontrivial to tell if the objective value is small because the solution is good in terms of the outer minimization, or because it is poor in terms of the inner maximization. We leave this issue as future work.

5.5 Theoretical Analysis

A theoretically appealing property of the proposed method is that its objective consists of simple sample averages. This enables us to establish a generalization error bound in terms of the Rademacher complexity (Koltchinskii, 2001; Mohri et al., 2012).

Denote $\varepsilon_G(f) := \sup_{g \in L^2(p)} J(f, g) - \sup_{g \in G} J(f, g)$. Also, let $\mathfrak{R}_q^N(H)$ denote the *Rademacher complexity* of a set of functions H over N random variables following probability

density q (refer to Appendix A.5 for the definition). Proofs of the following theorems and corollary can be found in Appendix A.5, Appendix A.6, and Appendix A.7.

Theorem 5.5.1. *Assume that* $n_1 = n_2$, $\widetilde{n}_1 = \widetilde{n}_2$, $p_1(\boldsymbol{x}) = p_2(\boldsymbol{x})$, $W := \inf_{x\in\mathcal{X}} |\mu_w(\boldsymbol{x})| > 0$, $M_{\mathcal{Z}} := \sup_{z\in\mathcal{Z}} |z| < \infty$, $M_F := \sup_{f\in F, x\in\mathcal{X}} |f(x)| < \infty$, *and* $M_G := \sup_{g\in G, x\in\mathcal{X}} |g(x)| < \infty$. *Then, the following holds with probability at least* $1-\delta$ *for every* $f \in F$*:*

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[(f(\boldsymbol{x}) - u(\boldsymbol{x}))^2] \leq \frac{1}{W^2}\left[\sup_{g\in G} \widehat{J}(f,g) + \mathcal{R}^{n,\widetilde{n}}_{F,G} + \left(\frac{M_z}{\sqrt{2n}} + \frac{M_w}{\sqrt{2\widetilde{n}}}\right)\sqrt{\log\frac{2}{\delta}} + \varepsilon_G(f)\right],$$

where $M_z := 4M_{\mathcal{Y}}M_{\mathcal{G}} + M_{\mathcal{G}}^2/2$, $M_w = 2M_{\mathcal{F}}M_{\mathcal{G}} + M_{\mathcal{G}}^2/2$, *and* $\mathcal{R}^{n,\widetilde{n}}_{F,G} := 2(M_F + 4M_Z)\mathfrak{R}^n_{p(\boldsymbol{x},z)}(G) + 2(2M_F + M_G)\mathfrak{R}^{\widetilde{n}}_{p(\boldsymbol{x},w)}(F) + 2(M_F + M_G)\mathfrak{R}^{\widetilde{n}}_{p(\boldsymbol{x},w)}(G)$.

In particular, the following bound holds for the linear-in-parameter models.

Corollary 5.5.1. *Let* $F = \{x \mapsto \boldsymbol{\alpha}^\top\boldsymbol{\phi}(\boldsymbol{x}) \mid \|\boldsymbol{\alpha}\|_2 \leq \Lambda_F\}$, $G = \{x \mapsto \boldsymbol{\beta}^\top\boldsymbol{\psi}(\boldsymbol{x}) \mid \|\boldsymbol{\beta}\|_2 \leq \Lambda_G\}$. *Assume that* $r_F := \sup_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{\phi}(\boldsymbol{x})\| < \infty$ *and* $r_G := \sup_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{\psi}(\boldsymbol{x})\| < \infty$, *where* $\|\cdot\|_2$ *is the* L_2*-norm. Under the assumptions of Theorem 5.5.1, it holds with probability at least* $1-\delta$ *that for every* $f \in F$*,*

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[(f(\boldsymbol{x}) - u(\boldsymbol{x}))^2] \leq \frac{1}{W^2}\left[\sup_{g\in G} \widehat{J}(f,g) + \frac{C_z\sqrt{\log\frac{2}{\delta}} + D_z}{\sqrt{2n}} + \frac{C_w\sqrt{\log\frac{2}{\delta}} + D_w}{\sqrt{2\widetilde{n}}} + \varepsilon_G(f)\right],$$

where $C_z := r_G^2\Lambda_G^2 + 4r_G\Lambda_G M_{\mathcal{Y}}$, $C_w := 2r_F^2\Lambda_F^2 + 2r_F r_G\Lambda_F\Lambda_G + r_G^2\Lambda_G^2$, $D_z := r_G^2\Lambda_G^2/2 + 4r_G\Lambda_G M_{\mathcal{Y}}$, *and* $D_w := r_G^2\Lambda_G^2/2 + 4r_F r_G\Lambda_F\Lambda_G$.

Theorem 5.5.1 and Corollary 5.5.1 imply that minimizing $\sup_{g\in G} \widehat{J}(f,g)$, as the proposed method does, amounts to minimizing an upper bound of the mean squared error. In fact, for the linear-in-parameter models, it can be shown that the mean squared error of the proposed estimator is upper bounded by $O(1/\sqrt{n} + 1/\sqrt{\widetilde{n}})$ plus some model mis-specification error with high probability as follows.

Theorem 5.5.2 (Informal)**.** *Let* $\widehat{f} \in F$ *be any approximate solution to* $\inf_{f\in F}\sup_{g\in G} \widehat{J}(f,g)$ *with sufficient precision. Under the assumptions of Corollary 5.5.1, it holds with probability at least* $1-\delta$ *that*

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[(\widehat{f}(\boldsymbol{x}) - u(\boldsymbol{x}))^2] \leq O\left(\left(\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{\widetilde{n}}}\right)\log\frac{1}{\delta}\right) + \frac{2\varepsilon_G^F + \varepsilon_F}{W^2}, \tag{5.13}$$

where $\varepsilon_G^F := \sup_{f\in F} \varepsilon_G(f)$ *and* $\varepsilon_F := \inf_{f\in F} J(f)$.

A more formal version of Theorem 5.5.2 can be found in Appendix A.7.

Note that the rate of convergence derived here may not be optimal and could be improved. Traditional asymptotic analyses show that of the *maximum-likelihood estimator* and the *M-estimators* converge to their estimands at rate of $O_p(1/\sqrt{n})$ under some regularity conditions (Vaart, 1998), and thus the mean squared error of a linear regressor of this

type typically tends to zero at $O_p(1/n)$, where n is the sample size, and $O_p(\cdot)$ denotes the rate of convergence in probability. More recent work has shown non-asymptotic excess risk bounds for the linear regression (Hsu et al., 2014) and for the general empirical risk minimization framework under various assumptions such as the low-noise condition for classification problems (Mammen and Tsybakov, 1999) and the strong convexity of the loss function (Bartlett et al., 2005; Koltchinskii, 2006).

However, our problem does not fit into those frameworks due to its min-max form. Further investigation in this direction is left as future work.

5.6 More General Loss Functions

Our framework can be extended to more general loss functions:

$$\inf_{f \in L^2(p)} \mathbf{E}[\ell(\mu_w(\boldsymbol{x}) f(\boldsymbol{x}), 2\mu_z(\boldsymbol{x}))], \tag{5.14}$$

where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function that is lower semi-continuous and convex with respect to both the first and the second arguments, where a function $\varphi : \mathbb{R} \to \mathbb{R}$ is *lower semi-continuous* if $\liminf_{y \to y_0} \varphi(y) = \varphi(y_0)$ for every $y_0 \in \mathbb{R}$ (Rockafellar, 1970).[*3] As with the squared loss, a major difficulty in solving this optimization problem is that the operand of the expectation has nonlinear dependency on both $\mu_w(\boldsymbol{x})$ and $\mu_z(\boldsymbol{x})$ at the same time. Below, we will show a way to transform the objective functional into a form that can be easily approximated using separately labeled samples.

From the assumptions on ℓ, we have $\ell(y, y') = \sup_{z \in \mathbb{R}} yz - \ell^*(z, y')$, where $\ell^*(\cdot, y')$ is the convex conjugate of the function $y \mapsto \ell(y, y')$ defined for any $y' \in \mathbb{R}$ as $z \mapsto \ell^*(z, y') = \sup_{y \in \mathbb{R}}[yz - \ell(y, y')]$ (see Rockafellar (1970)). Hence,

$$\mathbf{E}[\ell(\mu_w(\boldsymbol{x}) f(\boldsymbol{x}), 2\mu_z(\boldsymbol{x}))] = \sup_{g \in L^2(p)} \mathbf{E}[\mu_w(\boldsymbol{x}) f(\boldsymbol{x}) g(\boldsymbol{x}) - \ell^*(g(\boldsymbol{x}), 2\mu_z(\boldsymbol{x}))].$$

Similarly, we obtain $\mathbf{E}[\ell^*(g(\boldsymbol{x}), 2\mu_z(\boldsymbol{x}))] = \sup_{h \in L^2(p)} 2\,\mathbf{E}[\mu_z(\boldsymbol{x}) h(\boldsymbol{x})] - \mathbf{E}[\ell_*^*(g(\boldsymbol{x}), h(\boldsymbol{x}))]$, where $\ell_*^*(y, \cdot)$ is the convex conjugate of the function $y' \mapsto \ell^*(y, y')$ defined for any $y, z' \in \mathbb{R}$ by $\ell_*^*(y, z') := \sup_{y' \in \mathbb{R}}[y'z - \ell^*(y, y')]$. Thus, Eq. (5.14) can be rewritten as

$$\inf_{f \in L^2(p)} \sup_{g \in L^2(p)} \inf_{h \in L^2(p)} K(f, g, h),$$

where $K(f, g, h) := \mathbf{E}[\mu_w(\boldsymbol{x}) f(\boldsymbol{x}) g(\boldsymbol{x})] - 2\,\mathbf{E}[\mu_z(\boldsymbol{x}) h(\boldsymbol{x})] + \mathbf{E}[\ell_*^*(g(\boldsymbol{x}), h(\boldsymbol{x}))]$. Since μ_w and μ_z appear separately and linearly, $K(f, g, h)$ can be approximated by sample averages using separately labeled samples.

[*3] $\liminf_{y \to y_0} \varphi(y) := \lim_{\delta \searrow 0} \inf_{|y - y_0| \le \delta} \varphi(y)$.

5.7 Experiments

In this section, we test the proposed method and compare it with baselines.

5.7.1 Data Sets

We use the following data sets for experiments.

Synthetic data: Features $\boldsymbol{x}$ are drawn from the two-dimensional Gaussian distribution with mean zero and covariance $10\boldsymbol{I}_2$. We set $p(y \mid \boldsymbol{x}, t)$ as the following logistic models: $p(y \mid \boldsymbol{x}, t) = 1/(1 - \exp(-y\boldsymbol{a}_t^\top \boldsymbol{x}))$, where $\boldsymbol{a}_{-1} = (10, 10)^\top$ and $\boldsymbol{a}_1 = (10, -10)^\top$. We also use the logistic models for $p_k(t \mid \boldsymbol{x})$: $p_1(t \mid \boldsymbol{x}) = 1/(1 - \exp(-tx_2))$ and $p_2(t \mid \boldsymbol{x}) = 1/(1 - \exp(-t\{x_2 + b\}))$, where b is varied over 25 equally spaced points in $[0, 10]$. We investigate how the performance changes when the difference between $p_1(t \mid \boldsymbol{x})$ and $p_2(t \mid \boldsymbol{x})$ varies.

Email data: This data set consists of data collected in an email advertisement campaign for promoting customers to visit a website of a store (Hillstrom, 2008; Radcliffe, 2008). Outcomes are whether customers visited the website or not. We use 4×5000 and 2000 randomly sub-sampled data points for training and evaluation, respectively.

Jobs data: This data set consists of randomized experimental data obtained from a job training program called the National Supported Work Demonstration (LaLonde, 1986), available at `http://users.nber.org/~rdehejia/data/nswdata2.html`. There are 9 features, and outcomes are income levels after the training program. The sample sizes are 297 for the treatment group and 425 for the control group. We use 4×50 randomly sub-sampled data points for training and 100 for evaluation.

Criteo data: This data set consists of banner advertisement log data collected by Criteo (Lefortier et al., 2016) available at `http://www.cs.cornell.edu/~adith/Criteo/`. The task is to select a product to be displayed in a given banner so that the click rate will be maximized. We only use records for banners with only one advertisement slot. Each display banner has 10 features, and each product has 35 features. We take the 12th feature of a product as a treatment variable merely because it is a well-balanced binary variable. The outcome is whether the displayed advertisement was clicked. We treat the data set as the population although it is biased from the actual population since non-clicked impressions were randomly sub-sampled down to 10% to reduce the data set size. We made two subsets with different treatment policies by appropriately sub-sampling according to the predefined treatment policies (see Appendix A.12). We set $p_k(t \mid \boldsymbol{x})$ as $p_1(t \mid \boldsymbol{x}) = 1/(1 + \exp(-t\mathbf{1}^\top \boldsymbol{x}))$ and $p_2(t \mid \boldsymbol{x}) = 1/(1 + \exp(t\mathbf{1}^\top \boldsymbol{x}))$, where $\mathbf{1} := (1, \ldots, 1)^\top$.

5.7.2 Experimental Settings

We conduct experiments under the following settings.

Methods compared: We compare the proposed method with baselines that separately estimate the four conditional expectations in Eq. (5.6). In the case of binary outcomes,

we use the logistic-regression-based (denoted by FourLogistic) and a neural-network-based method trained with the soft-max cross-entropy loss (denoted by FourNNC). In the case of real-valued outcomes, the ridge-regression-based (denoted by FourRidge) and a neural-network-based method trained with the squared loss (denoted by FourNNR). The neural networks are fully connected ones with two hidden layers each with 10 hidden units. For the proposed method, we use the linear-in-parameter models with Gaussian basis functions centered at randomly sub-sampled training data points (see Appendix A.11 for more details).

Performance evaluation: We evaluate trained uplift models by the area under the uplift curve (AUUC) estimated on test samples with joint labels as well as *uplift curves* (Radcliffe and Surry, 1999). The uplift curve of an estimated individual uplift is the trajectory of the average uplift when individuals are gradually moved from the control group to the treated group in the descending order according to the ranking given by the estimated individual uplift. These quantities can be estimated when data are randomized experiment ones. The Criteo data are not randomized experiment data unlike other data sets, but there are accurately logged propensity scores available. In this case, uplift curves and the AUUCs can be estimated using the inverse propensity scoring (Austin, 2011; Li et al., 2012). We conduct 50 trials of each experiment with different random seeds.

5.7.3 Results

The results on the synthetic data are summarized in Figure 5.1. From the plots, we can see that all methods perform relatively well in terms of AUUCs when the policies are distant from each other (i.e., b is larger). However, the performance of the baseline methods immediately declines as the treatment policies get closer to each other (i.e., b is smaller).[*4] In contrast, the proposed method maintains its performance relatively longer until b reaches the point around 2. Note that the two policies would be identical when $b = 0$, which makes it impossible to identify the individual uplift from their samples by any method since the system in Eq. (5.5) degenerates. Figure 5.2 highlights their typical performances in terms of the squared error. For FourLogistic (Figure 5.2(a)) and FourNNC (Figure 5.2(b)), test points with small difference between the treatment probabilities, $|p_1(t = 1 \mid \boldsymbol{x}) - p_2(t = 1 \mid \boldsymbol{x})|$, tend to have very large estimation errors. On the other hand, the proposed method (Figure 5.2(c)) makes much small errors even for such points. More detailed plots including spatial information of test points are shown in Figure 5.3. Figure 5.4 shows results on real data sets. The proposed method and the baseline method with logistic regressors both performed better than the baseline method with neural nets on the Email data set (Figure 5.4(a)). On the Jobs data set, the proposed method again performed better than the baseline methods with neural networks. For the Criteo data set, the proposed method outperformed the baseline methods (Figure 5.4(c)). Overall, we confirmed the superiority of the proposed both on synthetic and real data sets.

[*4] The instability of performance of FourLogistic can be explained as follows. FourLogistic uses linear models, whose expressive power is limited. The resulting estimator has small variance with potentially large bias. Since different b induces different $u(\boldsymbol{x})$, the bias depends on b. For this reason, the method works well for some b but poorly for other b.

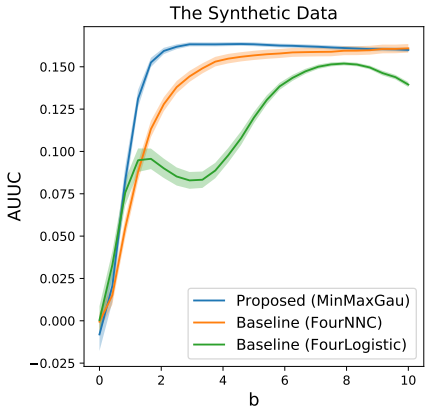
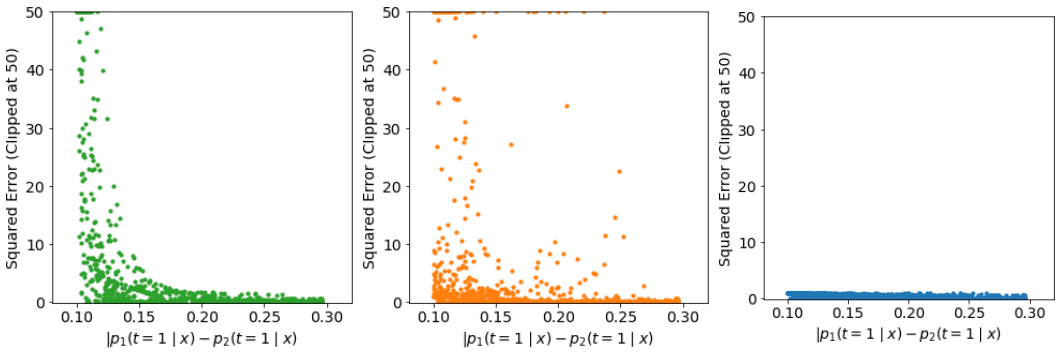


Figure 5.1: Results on the synthetic data. The plot shows the average AUUCs obtained by the proposed method and the baseline methods for different b. $p_1(t \mid \boldsymbol{x})$ and $p_2(t \mid \boldsymbol{x})$ are closer to each other when b is smaller.

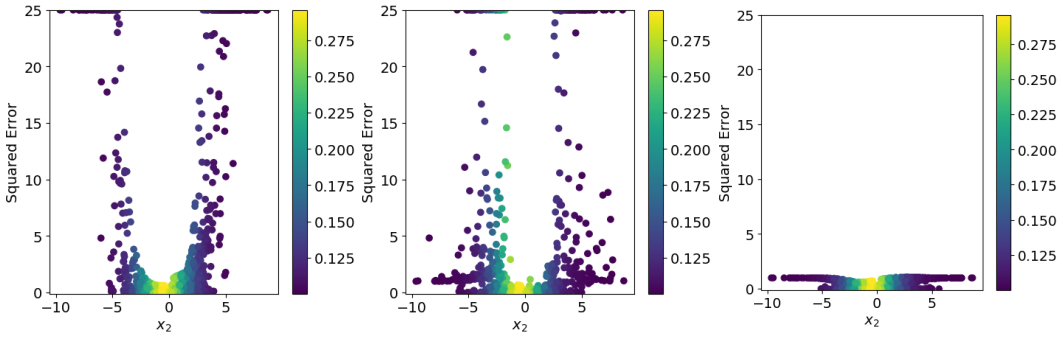


(A) Baseline (FourLogistic). (B) Baseline (FourNNC). (C) Proposed (MinMaxGau).

Figure 5.2: The scatter plots show the prediction errors of the estimated individual uplifts for the synthetic data with $b = 1$. Each point corresponds to a random test data point $\boldsymbol{x}$, whose vertical coordinate is $|p_1(t = 1 \mid \boldsymbol{x}) - p_2(t = 1 \mid \boldsymbol{x})|$, and the horizontal coordinate is the squared error of the prediction on $\boldsymbol{x}$. The errors are clipped down to 50 if they exceed it since some points are too large errors to be shown in the plots.

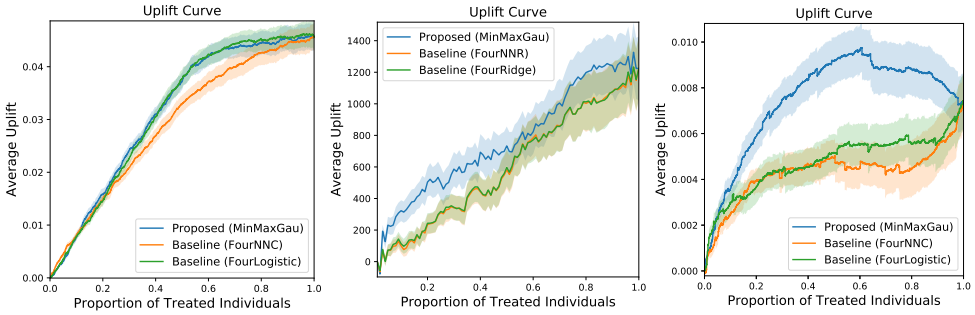
5.8 Conclusion

We proposed the first theoretically guaranteed method for a causal inference problem known as uplift modeling or individual treatment effect, and tested its performance through experiments on both synthetic and real datasets. The proposed objective is model-independent: we could use any models to approximate the individual uplift including ones tailored for specific problems and complex models such as neural networks. On the other hand, selecting the best models from candidates may be a challenging problem. This is because it may not be appropriate to select the one minimizing the proposed objective function since the objective value may be small just because the solution is poor in terms of the inner maximization. We proposed a theoretically guaranteed and practically useful method for uplift modeling or individual treatment effect estimation under the presence of systematic missing labels. The proposed method showed promising results in our experiments on synthetic and real data



(A) Baseline (FourLogistic). (B) Baseline (FourNNC). (C) Proposed (MinMaxGau).

Figure 5.3: The plots show the squared errors of the estimated individual uplifts on the synthetic data with $b = 1$. Each point is darker-colored when $|p_1(t = 1 \mid \boldsymbol{x}) - p_2(t = 1 \mid \boldsymbol{x})|$ is smaller, and lighter-colored otherwise. The errors are clipped down to 25 if they exceed it since some points are too large errors to be shown in the plots.



(A) The Email data. (B) The Jobs data. (C) The Criteo data.

Figure 5.4: Average uplifts as well as their standard errors on real-world data sets.

sets. The proposed framework is model-independent: any models can be used to approximate the individual uplift including ones tailored for specific problems and complex models such as neural networks. On the other hand, model selection may be a challenging problem due to the min-max structure. Our work considered the case of a binary treatment. However, there are many real-world problems where we have more than two kinds of treatments. The extension to multiple treatments is important future work. Addressing these issues would be important research directions to further improve the applicability and the performance of the proposed method.

Chapter 6

Conclusions and Future Work

This chapter concludes the thesis and discuss potential future research directions.

6.1 Conclusions

This thesis is devoted to investigation of learning from limited information. We categorized limitation of information into two types: quantitative limitation and qualitative limitation. As specific instances of learning from quantitatively limited information, we investigated multi-dimensional log-density gradient estimation and multi-task principal component analysis. As an instance of learning from qualitatively limited information, we studied uplift modeling from separate labels.

We proposed several approaches based on information sharing to these problems and developed practically useful learning algorithms. We demonstrated the effectiveness of the approaches through various experiments.

We summarize the contributions of this thesis below.

- **Information sharing between the output dimensions for multi-dimensional log-density gradient estimation from quantitatively limited information.** In Chapter 3, we proposed a method for multi-dimensional log-density gradient estimation from a limited amount of training data. Regarding each output dimension of the log-density gradient as a task, our method solves them simultaneously with a multi-task learning regularizer. For carefully designed models, the regularizer imposes the task relationships that generally hold in log-density gradient estimation regardless of the distribution, without strong assumptions.

- **Information sharing between projection matrices for multi-task principal component analysis from quantitatively limited information.** In Chapter 4, we proposed a method for solving multiple tasks of principal component analysis when each task has only a limited amount of training data. Our method uses a regularizer that makes projection matrices for different tasks close to each other to promote information sharing among them. The geometry-aware design of the regularizer based on the metric intrinsic with the manifold of projection matrices enables us to apply a

recent technique (Absil et al., 2009) for directory optimizing the projection matrices on the manifold.

- **Information sharing between slightly different populations for uplift modeling from qualitatively limited information.** In Chapter 5, we proposed a method for uplift modeling only from systematic missing labels called separate labels. We overcome the challenge that stems from this qualitatively limitation of supervision by sharing information between two different populations satisfying reasonable assumptions. Our proposed method uses all available information at once to directly estimate the learning target, leading to stable performance.

From these results, we conclude that information sharing is an important direction of research to pursue in order to realize machine learning from limited information.

6.2 Future Work

We discuss further extensions, investigations, and other future work of the thesis.

6.2.1 Application of the Regularizer of MT-LSLDG to Other Gradient-Related Problems

Although we focused on log-density gradient estimation in this thesis, the general idea of sharing information between partial derivatives could be applied to other problems involving estimation of gradients of functions.

For example, Fukumizu and Leng (2012); Suzuki and Sugiyama (2013); Tangkaratt et al. (2017) use estimated gradients of some statistical dependency measures for finding a subset or a combination of input features with best dependent on output variables by gradient methods. The estimation of the gradients is critical part of those algorithms, and our regularizer may be able to improve their performance.

6.2.2 Extension of MT-LSLDG to Estimation of Higher-Order Derivatives

Log-density gradient is the first-order derivative of the log-density. We could consider higher-order derivatives as its extensions. Sasaki et al. (2016, 2015) investigated estimation of the k-th order derivative of a density function for an arbitrary $k \in \mathbb{N}_+$. In this case the target function outputs a tensor of k modes. The design of our regularizer would naturally extend to such general cases by using higher-order derivatives of a common function as basis functions of linear-in-parameter models. However, we would have additional computational challenges to address because there would be more number of parameters.

6.2.3 Extension of MTLPCA to Multi-Task PCA with Other Task-Relatedness

We considered multi-task principal component analysis under the assumption that the optimal subspaces are similar to each other. Other types of relatedness between tasks may be considered as in previous work on multi-task learning. For example, Argyriou et al. (2008a) assumes that parameters for different tasks share a common subspace, Jacob et al. (2009a); Zhou et al. (2011) assumes that tasks are similar only within some clusters, and Lozano and Swirszcz (2012); Obozinski and Taskar (2006) considers situations where tasks share some sparsity structures These all consider unconstrained problems, where parameters are defined on the Euclidean space. Extensions of such work to a non-Euclidean space may not be straightforward but interesting directions to explore.

6.2.4 Extension of Uplift Modeling from Separate Labels to Multiple Treatments

Our uplift modeling method assumes that a treatment takes a binary value. It will be useful if we can extend it to cases where we have multiple treatments as was done in the standard setup (Rzepakowski and Jaroszewicz, 2012b). Although how to modify our direct estimator is not a trivial question to answer, the naive method can be easily extended to the general case, meaning that the general problem itself is feasible. Development of a method going beyond the (extended) naive method is an interesting and useful extension to work on in the future.

6.2.5 Extension of Uplift Modeling from Separate Labels to a Semi-Supervised Setting

We saw that uplift modeling is feasible even when we only have separate labels. In some applications, there may be standard joint labels available as well as separate labels. We may artificially convert jointly labeled samples to separately labeled samples as follows: We make two copies of each sample and remove the treatment label from one copy and the outcome label from the other one. However, this may not be optimal since we break the connection between the paired labels and throw away part of their information. It would be useful to develop a semi-supervised method that can fully utilize both types of samples.

6.2.6 Extension to Uplift Modeling from Further Limited Information

In binary classification, previous work has considered the situation where we only have positive samples and unlabeled samples while we have no negative samples (Blanchard et al., 2010; du Plessis et al., 2014, 2015; Elkan and Noto, 2008; Kiryo et al., 2017; Niu et al., 2016).

We may consider the similar situation in uplift modeling where outcomes are binary (positive and negative), and samples are labeled by the outcome only when they take the

positive value while samples are unlabeled otherwise, which limits supervised information in a different manner from that of separate labels. Investigating how to incorporate the techniques from the binary classification literature to uplift modeling is an interesting direction to proceed.

6.2.7 Deriving Fast Learning Rate for the Proposed Uplift Modeling Method

We showed an upper bound on the mean squared error for the proposed uplift modeling method. Our analysis relies on the general technique of the Rademacher complexity error bound, which does not consider specific properties of the objective function.

Our optimization problem originates from a least-squares problem, for which better upper bounds of rate $O(1/n)$ have been shown in the literature (Hsu et al., 2014; Vaart, 1998), n being the sample size. Similar results have been shown even for more general strongly convex objectives (Bartlett et al., 2005; Hsu et al., 2014; Koltchinskii, 2006; Mammen and Tsybakov, 1999; Vaart, 1998).

A natural question is whether a similar faster rate of convergence can be attained in our problem. However, it does not seem straightforward to answer this question because our problem takes the min-max form but not the standard form of minimization problem assumed by the techniques mentioned above. The approximation of the objective may affect the solutions to the outer minimization problem and the inner maximization at the same time, in a complex way. Development of analysis techniques for addressing this issue is also our future work.

Bibliography

Abrahams, E. and Silver, M. (2009). The Case for Personalized Medicine. *Journal of Diabetes Science and Technology*, 3(4):680–684.

Absil, P.-A., Mahony, R., and Sepulchre, R. (2009). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press.

Ando, R. K. and Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853.

Argyriou, A., Evgeniou, T., and Pontil, M. (2008a). Convex multi-task feature learning. *Machine Learning*, 73(3):243–272.

Argyriou, A., Evgeniou, T., and Pontil, M. (2008b). Convex multi-task feature learning. *Machine Learning*, 73(3):243–272.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Athey, S., Tibshirani, J., and Wager, S. (2016). Generalized Random Forests. *arXiv:1610.01271 [econ, stat]*. arXiv: 1610.01271.

Austin, P. C. (2011). An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate behavioral research*, 46(3):399–424.

Badeau, R., Richard, G., and David, B. (2008). Fast and stable YAST algorithm for principal and minor subspace tracking. *IEEE Transactions on Signal Processing*, 56(8):3437–3446.

Balzano, L., Nowak, R., and Recht, B. (2010). Online identification and tracking of subspaces from highly incomplete information. In *48th Annual Allerton Conference on Communication, Control, and Computing*, pages 704–711. IEEE.

Bandera, A., Pérez-Lorenzo, J. M., Bandera, J., and Sandoval, F. (2006). Mean shift based clustering of hough domain for fast line segment detection. *Pattern Recognition Letters*, 27(6):578–586.

Bartlett, P. L., Bousquet, O., and Mendelson, S. (2005). Local rademacher complexities. *The Annals of Statistics*, 33(4):1497–1537.

Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198.

Beran, R. (1976). Adaptive estimates for autoregressive processes. *Annals of the Institute of Statistical Mathematics*, 28(1):77–89.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554.

Bickel, S., Bogojeska, J., Lengauer, T., and Scheffer, T. (2008). Multi-task learning for HIV therapy screening. In *Proceedings of the 25th International Conference on Machine Learning*, pages 56–63. ACM.

Bishop, C. M. (2006a). *Pattern Recognition and Machine Learning*. Springer.

Bishop, C. M. (2006b). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Bishop, C. M. et al. (2006). *Pattern Recognition and Machine Learning*, volume 1. springer New York.

Blanchard, G., Lee, G., and Scott, C. (2010). Semi-Supervised Novelty Detection. *Journal of Machine Learning Research*, 11(Nov):2973–3009.

Boumal, N., Mishra, B., Absil, P.-A., and Sepulchre, R. (2014). Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459.

Burks, A. W. and Burks, A. R. (1981). First general-purpose electronic computer. *Annals of the History of Computing*, 3(4):310–389.

Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis? *Journal of the ACM*, 58(3):11.

Caponnetto, A., Micchelli, C. A., Pontil, M., and Ying, Y. (2008). Universal multi-task kernels. *The Journal of Machine Learning Research*, 9:1615–1646.

Cardoso, J.-F. and Souloumiac, A. (1996). Jacobi angles for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 17(1):161–164.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

Caruana, R. (1998). Multitask learning. In *Learning to Learn*, pages 95–133. Springer.

Catlett, J. (1991). *Inductive learning from subsets or disposal of excess training data considered harmful*. Basser Department of Computer Science. University of Sydney.

Chapelle, O., Schölkopf, B., and Zien, A. (2006). Semi-supervised learning.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799.

Chevallier, S., Barthélemy, Q., and Atif, J. (2013). Metrics for multivariate dictionaries. *arXiv preprint arXiv:1302.4242*.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.

Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition 2000*, volume 2, pages 142–149.

Cox, D. D. (1985). A penalty method for nonparametric estimation of the logarithmic derivative of a density function. *Annals of the Institute of Statistical Mathematics*, 37(1):271–288.

De Bie, T., Cristianini, N., and Rosipal, R. (2005). Eigenproblems in pattern recognition. In *Handbook of Geometric Computing*, pages 129–167. Springer.

Delac, K. and Grgic, M. (2004). A survey of biometric recognition methods. In *46th International Symposium on Electronics in Marine*, pages 184–193.

Devlaminck, D., Wyns, B., Grosse-Wentrup, M., Otte, G., and Santens, P. (2011). Multisubject learning for common spatial patterns in motor-imagery BCI. *Computational Intelligence and Neuroscience*, 2011:8.

Devlin, S. J., Gnanadesikan, R., and Kettenring, J. R. (1981). Robust estimation of dispersion matrices and principal components. *Journal of the American Statistical Association*, 76(374):354–362.

du Plessis, M. C., Niu, G., and Sugiyama, M. (2013). Clustering unclustered data: Unsupervised binary labeling of two datasets having different class balances. In *2013 Conference on Technologies and Applications of Artificial Intelligence*, pages 1–6.

du Plessis, M. C., Niu, G., and Sugiyama, M. (2014). Analysis of learning from positive and unlabeled data. In *Advances in Neural Information Processing Systems 27*, pages 703–711. Curran Associates, Inc.

du Plessis, M. C., Niu, G., and Sugiyama, M. (2015). Convex Formulation for Learning from Positive and Unlabeled Data. In *International Conference on Machine Learning*, pages 1386–1394.

du Plessis, M. C. and Sugiyama, M. (2014). Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks*, 50:110–119.

Edelman, A., Arias, T. A., and Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353.

Elkan, C. and Noto, K. (2008). Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 213–220, New York, NY, USA. ACM.

Evgeniou, T., Micchelli, C. A., and Pontil, M. (2005). Learning multiple tasks with kernel methods. *The Journal of Machine Learning Research*, 6:615–637.

Evgeniou, T. and Pontil, M. (2004a). Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117. ACM.

Evgeniou, T. and Pontil, M. (2004b). Regularized multi-task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117. ACM.

Feldman, V., Guruswami, V., Raghavendra, P., and Wu, Y. (2012). Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6):1558–1590.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

Flury, B. N. and Gautschi, W. (1986). An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form. *SIAM Journal on Scientific and Statistical Computing*, 7(1):169–184.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin.

Fukumizu, K. and Leng, C. (2012). Gradient-based kernel method for feature extraction and variable selection. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 2114–2122. Curran Associates, Inc.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40.

Gangwar, A. and Joshi, A. (2016). DeepIrisNet: Deep iris representation with applications in iris recognition and cross-sensor iris recognition. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2301–2305.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Graves, A., Mohamed, A., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

Gutierrez, P. and Gérardy, J.-Y. (2017). Causal inference and uplift modelling: A review of the literature. In *Proceedings of The 3rd International Conference on Predictive Applications and APIs*, volume 67 of *Proceedings of Machine Learning Research*, pages 1–13. PMLR.

Hachiya, H., Sugiyama, M., and Ueda, N. (2012). Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition. *Neurocomputing*, 80:93–101.

Hans-Hermann, B. (2008). Origins and extensions of the k-means algorithm in cluster analysis. *Journal Electronique d'Histoire des Probabilités et de la Statistique Electronic Journal for History of Probability and Statistics*, 4(2).

Harandi, M. T., Salzmann, M., and Hartley, R. (2014). From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices. In *European Conference on Computer Vision*, pages 17–32. Springer International Publishing.

Hartford, J., Lewis, G., Leyton-Brown, K., and Taddy, M. (2017). Deep IV: A Flexible Approach for Counterfactual Prediction. In *International Conference on Machine Learning*, pages 1414–1423.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. pages 1026–1034.

He, X. and Niyogi, P. (2004). Locality preserving projections. In Thrun, S., Saul, L. K., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 153–160. MIT Press.

Hebb, D. O. et al. (1949). The organization of behavior.

Hernán, M. A. and Robins, J. M. (2018). *Causal Inference*. Boca Raton: Chapman & Hall/CRC, forthcoming.

Hill, J. L. (2011). Bayesian Nonparametric Modeling for Causal Inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240.

Hillstrom, K. (2008). The minethatdata e-mail analytics and data mining challenge. `https://blog.minethatdata.com/2008/03/minethatdata-e-mail-analytics-and-data.html`.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Holland, P. W. (1986). Statistics and causal inference. *Journal of the American Statistical Association*, 81(396):945–960.

Horev, I., Yger, F., and Sugiyama, M. (2015). Geometry-aware principal component analysis for symmetric positive definite matrices. In *Proceedings of The 7th Asian Conference on Machine Learning*, pages 1–16.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Hsu, D., Kakade, S. M., and Zhang, T. (2014). Random Design Analysis of Ridge Regression. *Foundations of Computational Mathematics*, 14(3):569–600.

Huber, P. J. (1985). Projection pursuit. *The Annals of Statistics*, 13(2):435–475.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

Imai, K. and Ratkovic, M. (2013). Estimating treatment effect heterogeneity in randomized program evaluation. *The Annals of Applied Statistics*, 7(1):443–470.

Imbens, G. W. (2014). Instrumental Variables: An Econometrician's Perspective. *Statistical Science*, 29(3):323–358.

Jacob, L., Vert, J.-p., and Bach, F. R. (2009a). Clustered Multi-Task Learning: A Convex Formulation. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 745–752. Curran Associates, Inc.

Jacob, L., Vert, J.-P., and Bach, F. R. (2009b). Clustered multi-task learning: A convex formulation. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 745–752. Curran Associates, Inc.

Jafri, R. and Arabnia, H. R. (2009). A survey of face recognition techniques. *Journal of Information Processing Systems*, 5(2):41–68.

Jaskowski, M. and Jaroszewicz, S. (2012). Uplift modeling for clinical trial data. In *ICML Workshop on Clinical Data Analysis*.

Johansson, F. D., Shalit, U., and Sontag, D. (2016). Learning Representations for Counterfactual Inference. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 3020–3029, New York, NY, USA. JMLR.

Joliffe, I. (1986). *Principal Component Analysis*. Springer.

Katsanis, S. H., Javitt, G., and Hudson, K. (2008). A Case Study of Personalized Medicine. *Science*, 320(5872):53–54.

Kawakubo, H., du Plessis, M. C., and Sugiyama, M. (2016). Computationally efficient class-prior estimation under class balance change using energy distance. *IEICE Transactions on Information and Systems*, E99.D(1):176–186.

Kaya, H., Tüfekci, P., and Gürgen, F. S. (2012). Local and global learning methods for predicting power of a combined gas & steam turbine. In *Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering (ICETCEE'2012), Dubai*.

Kiryo, R., Niu, G., du Plessis, M. C., and Sugiyama, M. (2017). Positive-unlabeled learning with non-negative risk estimator. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 1675–1685. Curran Associates, Inc.

Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. M. (2009). Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181.

Koltchinskii, V. (2001). Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914.

Koltchinskii, V. (2006). Local rademacher complexities and oracle inequalities in risk minimization. *The Annals of Statistics*, 34(6):2593–2656.

Künzel, S. R., Sekhon, J. S., Bickel, P. J., and Yu, B. (2017). Meta-learners for Estimating Heterogeneous Treatment Effects using Machine Learning. *arXiv:1706.03461 [math, stat]*. arXiv: 1706.03461.

LaLonde, R. J. (1986). Evaluating the econometric evaluations of training programs with experimental data. *The American economic review*, pages 604–620.

Lefortier, D., Swaminathan, A., Gu, X., Joachims, T., and de Rijke., M. (2016). Large-scale validation of counterfactual learning methods: A test-bed. In *NIPS Workshop on "Inference and Learning of Hypothetical and Counterfactual Interventions in Complex Systems"*.

Lewis, G. and Syrgkanis, V. (2018). Adversarial Generalized Method of Moments. *arXiv:1803.07164 [cs, econ, math, stat]*. arXiv: 1803.07164.

Li, L., Chu, W., Langford, J., Moon, T., and Wang, X. (2012). An unbiased offline evaluation of contextual bandit algorithms with generalized linear models. In Glowacka, D., Dorard, L., and Shawe-Taylor, J., editors, *Proceedings of the Workshop on On-line Trading of*

Exploration and Exploitation 2, volume 26 of *Proceedings of Machine Learning Research*, pages 19–36, Bellevue, Washington, USA. PMLR.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web - WWW '10*, page 661, Raleigh, North Carolina, USA. ACM Press.

Li, L., Chu, W., Langford, J., and Wang, X. (2011). Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining - WSDM '11*, page 297, Hong Kong, China. ACM Press.

Lichman, M. (2013). UCI machine learning repository. URL `http://archive.ics.uci.edu/ml`.

Liu, S., Takeda, A., Suzuki, T., and Fukumizu, K. (2017). Trimmed density ratio estimation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4518–4528. Curran Associates, Inc.

Loftsgaarden, D. O. and Quesenberry, C. P. (1965). A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3):1049–1051.

Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., and Arnaldi, B. (2007). A review of classification algorithms for EEG-based brain–computer interfaces. *Journal of Neural Engineering*, 4(2):R1.

Lozano, A. C. and Swirszcz, G. (2012). Multi-level lasso for sparse multi-task regression. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 595–602. Omnipress.

Lu, N., Niu, G., Menon, A. K., and Sugiyama, M. (2018). On the minimal supervision for training any binary classifier from only unlabeled data. *arXiv:1808.10585 [cs, stat]*. arXiv: 1808.10585.

Lucas, D. D., Klein, R., Tannahill, J., Ivanova, D., Brandon, S., Domyancic, D., and Zhang, Y. (2013). Failure analysis of parameter-induced simulation crashes in climate models. *Geoscientific Model Development*, 6(4):1157–1171.

Ma, Y., Košecká, J., and Sastry, S. (2001). Optimization criteria and geometric algorithms for motion and structure estimation. *International Journal of Computer Vision*, 44(3):219–249.

Mammen, E. and Tsybakov, A. B. (1999). Smooth discrimination analysis. *The Annals of Statistics*, 27(6):1808–1829.

Menotti, D., Chiachia, G., Pinto, A., Schwartz, W. R., Pedrini, H., Falcão, A. X., and Rocha, A. (2015). Deep Representations for Iris, Face, and Fingerprint Spoofing Detection. *IEEE Transactions on Information Forensics and Security*, 10(4):864–879.

Micchelli, C. A. and Pontil, M. (2005a). On learning vector-valued functions. *Neural Computation*, 17(1):177–204.

Micchelli, C. A. and Pontil, M. (2005b). On learning vector-valued functions. 17(1):177–204.

Minsky, M. (1961). Steps toward Artificial Intelligence. *Proceedings of the IRE*, 49(1):8–30.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill series in computer science. McGraw-Hill, New York.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

Moore, B. (1981). Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32.

Moore, G. (1998). Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Naeem, M., Brunner, C., Leeb, R., Graimann, B., and Pfurtscheller, G. (2006). Seperability of four-class motor imagery data using independent components analysis. *Journal of Neural Engineering*, 3(3):208.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2010). Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861.

Niu, G., du Plessis, M. C., Sakai, T., Ma, Y., and Sugiyama, M. (2016). Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In *Advances in Neural Information Processing Systems 29*, pages 1199–1207. Curran Associates, Inc.

Obozinski, G. and Taskar, B. (2006). Multi-task feature selection. Technical report, The workshop of structural Knowledge Transfer for Machine Learning in the 23rd International Conference on Machine Learning (ICML).

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Patel, K., Han, H., and Jain, A. K. (2016). Secure face unlock: Spoof detection on smartphones. *IEEE Transactions on Information Forensics and Security*, 11(10):2268–2283.

Pearl, J. (2009). *Causality*. Cambridge university press.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Pérignon, C., Smith, D. R., and Villa, C. (2007). Why common factors in international bond returns are not so common. *Journal of International Money and Finance*, 26(2):284–304.

Radcliffe, N. (2007). Using control groups to target on predicted lift: Building and assessing uplift model. *Direct Marketing Analytics Journal*, pages 14–21.

Radcliffe, N. and Surry, P. (1999). Differential Response Analysis: Modeling True Responses by Isolating the Effect of a Single Action. *Credit Scoring and Credit Control IV*.

Radcliffe, N. J. (2008). Hillstrom's minethatdata email analytics challenge: An approach using uplift modelling. *Technical report, Stochastic Solutions Limited*.

Radcliffe, N. J. and Surry, P. D. (2011). Real-world uplift modelling with significance-based uplift trees. *White Paper TR-2011-1, Stochastic Solutions*.

Rätsch, G., Onoda, T., and Müller, K. R. (2001). Soft margins for adaboost. *Machine Learning*, 42(3):287–320.

Renault, R. (2015). Chapter 4 - Advertising in Markets. In Anderson, S. P., Waldfogel, J., and Str omberg, D., editors, *Handbook of Media Economics*, volume 1 of *Handbook of Media Economics*, pages 121–204. North-Holland.

Rockafellar, R. T. (1970). *Convex Analysis*. Princeton University Press.

Rosenbaum, P. R. (2010). *Design of observational studies*. Springer series in statistics. Springer, New York.

Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837.

Rubin, D. B. (2005). Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331.

Rzepakowski, P. and Jaroszewicz, S. (2012a). Decision trees for uplift modeling with single and multiple treatments. *Knowledge and Information Systems*, 32(2):303–327.

Rzepakowski, P. and Jaroszewicz, S. (2012b). Decision trees for uplift modeling with single and multiple treatments. *Knowledge and Information Systems*, 32(2):303–327.

Sakai, T., du Plessis, M. C., Niu, G., and Sugiyama, M. (2017). Semi-supervised classification based on classification from positive and unlabeled data. In *International Conference on Machine Learning*, pages 2998–3006.

Sakai, T., Niu, G., and Sugiyama, M. (2018). Semi-supervised AUC optimization based on positive-unlabeled learning. *Mach. Learn.*, 107(4):767–794.

Samek, W., Meinecke, F. C., and Müller, K.-R. (2013). Transferring subspaces between subjects in brain-computer interfacing. *IEEE Transactions on Biomedical Engineering*, 60(8):2289–2298.

Sasaki, H., Hyvärinen, A., and Sugiyama, M. (2014). Clustering via mode seeking by direct estimation of the gradient of a log-density. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2014)*, pages 19–34, Nancy, France.

Sasaki, H., Noh, Y.-K., Niu, G., and Sugiyama, M. (2016). Direct Density Derivative Estimation. *Neural Computation*, 28(6):1101–1140.

Sasaki, H., Noh, Y.-K., and Sugiyama, M. (2015). Direct Density-Derivative Estimation and Its Application in KL-Divergence Approximation. In *Artificial Intelligence and Statistics*, pages 809–818.

Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, Cambridge.

Shalit, U., Johansson, F. D., and Sontag, D. (2017). Estimating individual treatment effect: generalization bounds and algorithms. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3076–3085, International Convention Centre, Sydney, Australia. PMLR.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Singh, R. S. (1977). Applications of estimators of a density and its derivatives to certain statistical problems. *Journal of the Royal Statistical Society. Series B*, 39(3):357–363.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.

Sugiyama, M. and Kawanabe, M. (2012). *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press.

Sugiyama, M., Niu, G., Yamada, M., Kimura, M., and Hachiya, H. (2014). Information-maximization clustering based on squared-loss mutual information. *Neural Computation*, 26(1):84–131.

Sugiyama, M., Suzuki, T., and Kanamori, T. (2012). *Density Ratio Estimation in Machine Learning*. Cambridge University Press.

Sugiyama, M., Takeuchi, I., Suzuki, T., Kanamori, T., Hachiya, H., and Okanohara, D. (2010). Conditional density estimation via least-squares density ratio estimation. In *International Conference on Artificial Intelligence and Statistics*, pages 781–788.

Suzuki, T. and Sugiyama, M. (2013). Sufficient dimension reduction via squared-loss mutual information estimation. *Neural Computation*, 25(3):725–758.

Tangkaratt, V., Sasaki, H., and Sugiyama, M. (2017). Direct estimation of the derivative of quadratic mutual information with application in supervised dimension reduction. *Neural Computation*, 29(8):2076–2122.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, pages 640–646.

Tüfekci, P. (2014). Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, 60:126–140.

Tufféry, S. (2011). *Data mining and statistics for decision making*, volume 2. Wiley Chichester.

Vaart, A. W. v. d. (1998). *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Vapnik, V., Golowich, S. E., and Smola, A. (1997). Support vector method for function approximation, regression estimation, and signal processing. In *Advances in Neural Information Processing Systems*, pages 281–287. MORGAN KAUFMANN PUBLISHERS.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

Wager, S. and Athey, S. (2015). Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *arXiv:1510.04342 [math, stat]*.

Wang, B., Pineau, J., and Balle, B. (2016). Multitask generalized eigenvalue program. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2115–2121.

Wang, H., Banerjee, A., and Boley, D. (2011). Common component analysis for multiple covariance matrices. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 956–964. ACM.

Wang, X., Zhang, C., and Zhang, Z. (2009). Boosted multi-task learning for face verification with applications to web image and video search. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2009*, pages 142–149. IEEE.

Warga, J. (1963). Minimizing certain convex functions. *Journal of the Society for Industrial & Applied Mathematics*, 11(3):588–593.

Warmuth, M. K. and Kuzmin, D. (2007). Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. In Schölkopf, B., Platt, J. C., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 1481–1488. MIT Press.

Wasserman, L. (2013). *All of statistics: a concise course in statistical inference*. Springer Science & Business Media.

Yamada, M., Suzuki, T., Kanamori, T., Hachiya, H., and Sugiyama, M. (2013). Relative density-ratio estimation for robust distribution comparison. *Neural Computation*, 25(5):1324–1370.

Yger, F. (2013). A review of kernels on covariance matrices for BCI applications. In *2013 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE.

Yger, F., Lotte, F., and Sugiyama, M. (2015). Averaging covariance matrices for EEG signal classification based on the CSP: an empirical study. In *23rd European Signal Processing Conference*, pages 2721–2725. IEEE.

Zhang, L., Dong, W., Zhang, D., and Shi, G. (2010). Two-stage image denoising by principal component analysis with local pixel grouping. *Pattern Recognition*, 43(4):1531–1549.

Zhang, Y. (2013). Heterogeneous-neighborhood-based multi-task local learning algorithms. In *Advances in Neural Information Processing Systems*, pages 1896–1904.

Zhang, Y. and Yeung, D.-Y. (2011). Multi-task learning in heterogeneous feature spaces. In *Proceedings of the National Conference on Artificial Intelligence*.

Zhou, J., Chen, J., and Ye, J. (2011). Clustered multi-task learning via alternating structure optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 702–710. Curran Associates, Inc.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286.

Appendix

A Supplementary Material for Uplift Modeling from Separate Labels

This section is for supplementary materials for Chapter 5.

A.1 Average Uplift in Terms of the Individual Uplift

$$
\begin{aligned}
U(\pi) &= \iint \sum_{t=-1,1} yp(y \mid t, \boldsymbol{x})\pi(t \mid \boldsymbol{x})p(\boldsymbol{x})\mathrm{d}y\mathrm{d}\boldsymbol{x} - \iint \sum_{t=-1,1} yp(y \mid t, \boldsymbol{x})1[t=-1]p(\boldsymbol{x})\mathrm{d}y\mathrm{d}\boldsymbol{x} \\
&= \iint y[p(y \mid t=1, \boldsymbol{x})\pi(t=1 \mid \boldsymbol{x}) - p(y \mid t=-1, \boldsymbol{x})\pi(t=1 \mid \boldsymbol{x})]p(\boldsymbol{x})\mathrm{d}y\mathrm{d}\boldsymbol{x} \\
&= \iint y[p(y \mid t=1, \boldsymbol{x}) - p(y \mid t=-1, \boldsymbol{x})]\pi(t=1 \mid \boldsymbol{x})p(\boldsymbol{x})\mathrm{d}y\mathrm{d}\boldsymbol{x} \\
&= \int u(\boldsymbol{x})\pi(t=1 \mid \boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \qquad (6.1)
\end{aligned}
$$

A.2 Area Under the Uplift Curve and Ranking

Define the following symbols:

- $C_\alpha := \Pr[f(\boldsymbol{x}) < \alpha]$,
- $U(\alpha; f) := \int u(\boldsymbol{x})1[\alpha \leq f(\boldsymbol{x})]p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$,
- $\mathrm{Rank}(f) := \mathbf{E}[1[f(\boldsymbol{x}') \leq f(\boldsymbol{x})][u(\boldsymbol{x}') - u(\boldsymbol{x})]]$,
- $\mathrm{AUUC}(f) := \int_0^1 U(\alpha; f)\mathrm{d}C_\alpha$.

Then, we have

$$
\begin{aligned}
\mathrm{AUUC}(f) &= \int_{-\infty}^{\infty} U(\alpha)\frac{\mathrm{d}C_\alpha}{\mathrm{d}\alpha}\mathrm{d}\alpha \\
&= \int_{-\infty}^{\infty} U(\alpha)p_{f(\boldsymbol{x})}(\alpha)\mathrm{d}\alpha \\
&= \int_{\mathbb{R}^d} U(f(\boldsymbol{x}))p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\
&= \iint 1[f(\boldsymbol{x}) \leq f(\boldsymbol{x}')]u(\boldsymbol{x}')p(\boldsymbol{x}')\mathrm{d}\boldsymbol{x}'p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\
&= \mathbf{E}[1[f(\boldsymbol{x}) \leq f(\boldsymbol{x}')]u(\boldsymbol{x}')] \\
&(= \mathbf{E}[1[f(\boldsymbol{x}) \leq f(\boldsymbol{x}')][y^+ - y^-]]),
\end{aligned}
$$

where $y^+ \sim p(y \mid \boldsymbol{x}', t = 1)$ and $y^- \sim p(y \mid \boldsymbol{x}', t = -1)$.

Assuming $\Pr[f(\boldsymbol{x}') = f(\boldsymbol{x})] = 0$, we have

$$\begin{aligned}
\mathrm{Rank}(f) &:= \mathbf{E}[1[f(\boldsymbol{x}) \geq f(\boldsymbol{x}')][u(\boldsymbol{x}) - u(\boldsymbol{x}')]] \\
&= \mathbf{E}[1[f(\boldsymbol{x}) \geq f(\boldsymbol{x}')]u(\boldsymbol{x})] \\
&\quad - \mathbf{E}[1[f(\boldsymbol{x}) \geq f(\boldsymbol{x}')]u(\boldsymbol{x}')] \\
&= \mathrm{AUUC}(f) - \mathbf{E}[(1 - 1[f(\boldsymbol{x}) \leq f(\boldsymbol{x}')])u(\boldsymbol{x}')] \\
&= \mathbf{E}[u(\boldsymbol{x})] - 2\,\mathrm{AUUC}(f).
\end{aligned}$$

Thus, $\mathrm{Rank}(f) = 2(\mathrm{AUUC}(f) - \mathrm{AUUC}(r))$, where $r : \mathbb{R}^d \to \mathbb{R}$ is the random ranking scoring function that outputs 1 or -1 with probability $1/2$ for any input $\boldsymbol{x}$. $\mathrm{Rank}(f)$ is maximized when $f(\boldsymbol{x}) \leq f(\boldsymbol{x}') \iff u(\boldsymbol{x}) \leq u(\boldsymbol{x}')$ for almost every pair of $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{x} \in \mathbb{R}^d$. In particular, $f = u$ is a maximizer of the objective.

A.3 Proof of Lemma 5.4.1

Lemma 5.4.1. For every $\boldsymbol{x}$ such that $p_1(\boldsymbol{x}) \neq p_2(\boldsymbol{x})$, $u(\boldsymbol{x})$ can be expressed as

$$u(\boldsymbol{x}) = 2 \times \frac{\mathbf{E}_{y \sim p_1(y|\boldsymbol{x})}[y] - \mathbf{E}_{y \sim p_2(y|\boldsymbol{x})}[y]}{\mathbf{E}_{t \sim p_1(t|\boldsymbol{x})}[t] - \mathbf{E}_{t \sim p_2(t|\boldsymbol{x})}[t]}. \tag{6.2}$$

Proof.

$$\begin{aligned}
\mathop{\mathbf{E}}_{y \sim p_1(y|\boldsymbol{x})}[y] - \mathop{\mathbf{E}}_{y \sim p_2(y|\boldsymbol{x})}[y] &= \int \sum_{\tau=-1,1} y p(y \mid \boldsymbol{x}, t = \tau) p_1(t = \tau \mid \boldsymbol{x}) \mathrm{d}y \\
&\quad - \int \sum_{\tau=-1,1} y p(y \mid \boldsymbol{x}, t = \tau) p_2(t = \tau \mid \boldsymbol{x}) \mathrm{d}y \\
&= \int \sum_{\tau=-1,1} y p(y \mid \boldsymbol{x}, t = \tau)(p_1(t = \tau \mid \boldsymbol{x}) - p_2(t = \tau \mid \boldsymbol{x})) \mathrm{d}y \\
&= \sum_{\tau=-1,1} \mathop{\mathbf{E}}_{y \sim p(y|\boldsymbol{x}, t=\tau)}[y](p_1(t = \tau \mid \boldsymbol{x}) - p_2(t = \tau \mid \boldsymbol{x})) \\
&= \mathop{\mathbf{E}}_{y \sim p(y|\boldsymbol{x}, t=1)}[y](p_1(t = 1 \mid \boldsymbol{x}) - p_2(t = 1 \mid \boldsymbol{x})) \\
&\quad + \mathop{\mathbf{E}}_{y \sim p(y|\boldsymbol{x}, t=-1)}[y](1 - p_1(t = 1 \mid \boldsymbol{x}) - 1 + p_2(t = 1 \mid \boldsymbol{x})) \\
&= u(\boldsymbol{x})(p_1(t = 1 \mid \boldsymbol{x}) - p_2(t = 1 \mid \boldsymbol{x})).
\end{aligned}$$

When $p_1(t = 1 \mid \boldsymbol{x}) \neq p_2(t = 1 \mid \boldsymbol{x})$, it holds that

$$\begin{aligned}
u(\boldsymbol{x}) &= \frac{\mathbf{E}_{y \sim p_1(y|\boldsymbol{x})}[y] - \mathbf{E}_{y \sim p_2(y|\boldsymbol{x})}[y]}{p_1(t = 1 \mid \boldsymbol{x}) - p_2(t = 1 \mid \boldsymbol{x})} \\
&= 2 \times \frac{\mathbf{E}_{y \sim p_1(y|\boldsymbol{x})}[y] - \mathbf{E}_{y \sim p_2(y|\boldsymbol{x})}[y]}{\mathbf{E}_{t \sim p_1(t|\boldsymbol{x})}[t] - \mathbf{E}_{t \sim p_2(t|\boldsymbol{x})}[t]}.
\end{aligned}$$

□

A.4 Proof of Lemma 5.4.2

Lemma 5.4.2. For every $\boldsymbol{x}$ such that $p_1(\boldsymbol{x}) \neq p_2(\boldsymbol{x})$, $u(\boldsymbol{x})$ can be expressed as

$$u(\boldsymbol{x}) = 2 \times \frac{\mathbf{E}[z \mid \boldsymbol{x}]}{\mathbf{E}[w \mid \boldsymbol{x}]},$$

where $\mathbf{E}[z \mid \boldsymbol{x}]$ and $\mathbf{E}[w \mid \boldsymbol{x}]$ are the conditional expectations of z given $\boldsymbol{x}$ over $p(z \mid \boldsymbol{x})$ and w given $\boldsymbol{x}$ over $p(w \mid \boldsymbol{x})$, respectively.

Proof. We have

$$\begin{aligned}
\mathbf{E}[z \mid \boldsymbol{x}] &= \int \zeta \left[\frac{1}{2} p_1(y = \zeta \mid \boldsymbol{x}) + \frac{1}{2} p_2(y = -\zeta \mid \boldsymbol{x})\right] \mathrm{d}\zeta \\
&= \frac{1}{2} \int \zeta p_1(y = \zeta \mid \boldsymbol{x}) \mathrm{d}\zeta + \frac{1}{2} \int \zeta p_2(y = -\zeta \mid \boldsymbol{x}) \mathrm{d}\zeta \\
&= \frac{1}{2} \int y p_1(y \mid \boldsymbol{x}) \mathrm{d}y - \frac{1}{2} \int y p_2(y \mid \boldsymbol{x}) \mathrm{d}y \\
&= \frac{1}{2} \mathop{\mathbf{E}}_{y \sim p_1(y \mid \boldsymbol{x})}[y] - \frac{1}{2} \mathop{\mathbf{E}}_{y \sim p_2(y \mid \boldsymbol{x})}[y].
\end{aligned}$$

Similarly, we obtain

$$\mathbf{E}[w \mid \boldsymbol{x}] = \frac{1}{2} \mathop{\mathbf{E}}_{t \sim p_1(t \mid \boldsymbol{x})}[t] - \frac{1}{2} \mathop{\mathbf{E}}_{t \sim p_2(t \mid \boldsymbol{x})}[t].$$

Thus,

$$2 \times \frac{\mathbf{E}[z \mid \boldsymbol{x}]}{\mathbf{E}[w \mid \boldsymbol{x}]} = 2 \times \frac{\mathbf{E}_{y \sim p_1(y \mid \boldsymbol{x})}[y] - \mathbf{E}_{y \sim p_2(y \mid \boldsymbol{x})}[y]}{\mathbf{E}_{t \sim p_1(t \mid \boldsymbol{x})}[t] - \mathbf{E}_{t \sim p_2(t \mid \boldsymbol{x})}[t]} = u(\boldsymbol{x}).$$

□

A.5 Proof of Theorem 5.5.1

We restate Theorem 5.5.1 below.

Theorem 5.5.1. Assume that $n_1 = n_2$, $\widetilde{n}_1 = \widetilde{n}_2$, $p_1(\boldsymbol{x}) = p_2(\boldsymbol{x})$, $W := \inf_{x \in \mathcal{X}} |\mu_w(\boldsymbol{x})| > 0$, $M_{\mathcal{Z}} := \sup_{z \in \mathcal{Z}} |z| < \infty$, $M_F := \sup_{f \in F, x \in \mathcal{X}} |f(x)| < \infty$, and $M_G := \sup_{g \in G, x \in \mathcal{X}} |g(x)| < \infty$. Then, the following holds with probability at least $1 - \delta$ that for every $f \in F$,

$$\mathop{\mathbf{E}}_{\boldsymbol{x} \sim p(\boldsymbol{x})}[(f(\boldsymbol{x}) - u(\boldsymbol{x}))^2] \leq \frac{1}{W^2} \left[\sup_{g \in G} \widehat{J}(f, g) + \mathcal{R}^{n, \widetilde{n}}_{F, G} + \left(\frac{M_z}{\sqrt{2n}} + \frac{M_w}{\sqrt{2\widetilde{n}}}\right) \sqrt{\log \frac{2}{\delta}} + \varepsilon_G(f)\right],$$

where $M_z := 4 M_{\mathcal{Y}} M_{\mathcal{G}} + M_{\mathcal{G}}^2/2$, $M_w = 2 M_{\mathcal{F}} M_{\mathcal{G}} + M_{\mathcal{G}}^2/2$, $\mathcal{R}^{n, \widetilde{n}}_{F, G} := 2(M_F + 4M_Z)\mathfrak{R}^n_{p(\boldsymbol{x}, z)}(G) + 2(2M_F + M_G)\mathfrak{R}^{\widetilde{n}}_{p(\boldsymbol{x}, w)}(F) + 2(M_F + M_G)\mathfrak{R}^{\widetilde{n}}_{p(\boldsymbol{x}, w)}(G)$.

Define $J(f, g)$ and $\widehat{J}(f, g)$ as in Section 3.2 and denote

$$\varepsilon_G(f) := \sup_{g \in L^2(p)} J(f, g) - \sup_{g \in G} J(f, g).$$

Definition A.1 (Rademacher Complexity)**.** We define the *Rademacher complexity* of H over N random variables following probability distribution q by

$$\mathfrak{R}_p^N(H) = \mathop{\mathbf{E}}_{V_1,\ldots,V_N,\sigma_1,\ldots,\sigma_N}\left[\sup_{h\in H}\frac{1}{N}\sum_{i=1}^{N}\sigma_i h(V_i)\right],$$

where $\sigma_1,\ldots,\sigma_N$ are independent, $\{-1,1\}$-valued uniform random variables.

Lemma A.1. *Under the assumptions of Theorem 5.5.1, with probability at least* $1-\delta$*, it holds that for every* $f \in F$*,*

$$J(f,g) \le \widehat{J}(f,g) + \mathfrak{R}_{F,G} + \left(\frac{M_z}{\sqrt{n}} + \frac{M_w}{\sqrt{\widetilde{n}}}\right)\sqrt{\log\frac{2}{\delta}}.$$

To prove Lemma A.1, we use the following lemma, which is a slightly modified version of Theorem 3.1 in Mohri et al. (2012).

Lemma A.2. *Let* H *be a set of real-valued functions on a measurable space* $\mathcal{D}$*. Assume that* $M := \sup_{h\in H, v\in\mathcal{D}} h(v) < \infty$*. Then, for any* $h \in H$ *and any* $\mathcal{D}$*-valued i.i.d. random variables* $V, V_1, \ldots, V_N$ *following density* q*, we have*

$$\mathbf{E}[h(V)] \le \frac{1}{N}\sum_{i=1}^{N} h(V_i) + 2\mathfrak{R}_q^N(H) + \sqrt{\frac{M^2}{N}\log\frac{1}{\delta}}. \tag{6.3}$$

Proof of Lemma A.2. We follow the proof of Theorem 3.1 in Mohri et al. (2012) except that we set the constant B_ϕ in Eq. (6.14) to M/m when we apply McDiarmid's inequality (Section A.13). □

Now, we prove Lemma A.1.

Proof of Lemma A.1. For any $f \in \mathcal{F}$, $g \in \mathcal{G}$, $\boldsymbol{x}', \widetilde{\boldsymbol{x}}' \in \mathcal{X}$, $z' \in \mathcal{Z} := \{y, -y \mid y \in \mathcal{Y}\}$, and $w' \in \{-1,1\}$, we define h_z and h_w as follows:

$$h_z(\boldsymbol{x}', z'; g) := -4z'g(\boldsymbol{x}') - \frac{1}{2}g(\boldsymbol{x}')^2,$$
$$h_w(\widetilde{\boldsymbol{x}}', w'; f, g) := w'f(\widetilde{\boldsymbol{x}}')g(\widetilde{\boldsymbol{x}}') - \frac{1}{2}g(\widetilde{\boldsymbol{x}}')^2.$$

Denoting $H_z := \{(\boldsymbol{x}', z') \mapsto h_z(\boldsymbol{x}', z'; g) \mid g \in G\}$, we have

$$\sup_{h\in H_z, \boldsymbol{x}'\in\mathcal{X}, z'\in\mathcal{Z}} |h(\boldsymbol{x}', z')| \le 4M_Z M_G + \frac{1}{2}M_G^2 =: M_z < \infty,$$

and thus, we can apply Lemma A.2 to confirm that with probability at least $1-\delta/2$,

$$\mathop{\mathbf{E}}_{(\boldsymbol{x},z)\sim p(\boldsymbol{x},z)}[h_z(\boldsymbol{x}, z; g)] \le \frac{1}{n}\sum_{(\boldsymbol{x}_i,z_i)\in S_z} h_z(\boldsymbol{x}_i, z_i; g) + 2\mathfrak{R}_p^n(H_z) + \sqrt{\frac{M_z^2}{n}\log\frac{2}{\delta}},$$

where $\{(\boldsymbol{x}_i, z_i)\}_{i=1}^{n} =: S_z$ are the samples defined in Section 5.4.1. Similarly, it holds that with probability at least $1 - \delta/2$,

$$\mathop{\mathbf{E}}_{(\widetilde{\boldsymbol{x}},w)\sim p(\boldsymbol{x},w)}[h_w(\widetilde{\boldsymbol{x}}, w; f, g)] \leq \frac{1}{\widetilde{n}} \sum_{(\widetilde{\boldsymbol{x}},w_i)\in S_w} h_w(\widetilde{\boldsymbol{x}}_i, w_i; f, g) + 2\mathfrak{R}_p^{\widetilde{n}}(H_w) + \sqrt{\frac{M_w^2}{\widetilde{n}} \log \frac{2}{\delta}},$$

where $H_w := \{(\widetilde{\boldsymbol{x}}', w') \mapsto h_w(\widetilde{\boldsymbol{x}}', w'; f, g) \mid f \in F, g \in G\}$, $M_w := M_F M_G + M_G^2/2$, and $\{(\widetilde{\boldsymbol{x}}_i, w_i)\}_{i=1}^{n} =: S_w$ are the samples defined in Section 5.4.1. By the union bound, we have the following with probability at least $1 - \delta$:

$$\mathop{\mathbf{E}}_{(\boldsymbol{x},z)\sim p(\boldsymbol{x},z)}[h_z(\boldsymbol{x}, z; g)] + \mathop{\mathbf{E}}_{(\widetilde{\boldsymbol{x}},w)\sim p(\boldsymbol{x},w)}[h_w(\widetilde{\boldsymbol{x}}, w; f, g)] \tag{6.4}$$

$$\leq \frac{1}{n} \sum_{(\boldsymbol{x}_i,z_i)\in S_z} h_z(\boldsymbol{x}_i, z_i, g) + \frac{1}{\widetilde{n}} \sum_{(\widetilde{\boldsymbol{x}},w_i)} h_w(\boldsymbol{x}_i, w_i, f, g) \tag{6.5}$$

$$+ 2(\mathfrak{R}_p^n(H_z) + \mathfrak{R}_p^{\widetilde{n}}(H_w)) + \left(\frac{M_z}{\sqrt{n}} + \frac{M_w}{\sqrt{\widetilde{n}}}\right)\sqrt{\log \frac{2}{\delta}}, \tag{6.6}$$

Using some properties of the Rademacher complexity including Talagrand's lemma, we can show that

$$\mathfrak{R}_p^n(H_z) \leq (M_F + 4M_Z)\mathfrak{R}_p^n(G), \tag{6.7}$$

$$\mathfrak{R}_p^{\widetilde{n}}(H_w) \leq (2M_F + M_G)\mathfrak{R}_p^{\widetilde{n}}(F) + (M_F + M_G)\mathfrak{R}_p^{\widetilde{n}}(G). \tag{6.8}$$

Clearly,

$$\widehat{J}(f, g) = \frac{1}{n} \sum_{(\boldsymbol{x}_i,z_i)\in S_z} h(\boldsymbol{x}_i, z_i; g) + \frac{1}{\widetilde{n}} \sum_{(\widetilde{\boldsymbol{x}}_i,w_i)\in S_w} h(\widetilde{\boldsymbol{x}}_i, w_i; f, g),$$

$$J(f, g) = \mathop{\mathbf{E}}_{(\boldsymbol{x},z)\sim p(\boldsymbol{x},z)}[h_z(\boldsymbol{x}, z; g)] + \mathop{\mathbf{E}}_{(\widetilde{\boldsymbol{x}},w)\sim p(\boldsymbol{x},z)}[h_w(\widetilde{\boldsymbol{x}}, w; f, g)].$$

From Eq. (6.6), Eq. (6.7), and Eq. (6.8), we obtain

$$J(f, g) \leq \widehat{J}(f, g) + \mathfrak{R}_{F,G} + \left(\frac{M_z}{\sqrt{n}} + \frac{M_w}{\sqrt{\widetilde{n}}}\right)\sqrt{\log \frac{2}{\delta}}, \tag{6.9}$$

where

$$\mathfrak{R}_{F,G} := 2(M_F + 4M_Z)\mathfrak{R}_p^n(G) + 2(2M_F + M_G)\mathfrak{R}_p^{\widetilde{n}}(F) + 2(M_F + M_G)\mathfrak{R}_p^{\widetilde{n}}(G).$$

□

Finally, we prove Theorem 5.5.1.

Proof of Theorem 5.5.1. From Lemma A.1, with probability at least $1-\delta$, it holds that for all $f \in F$

$$\sup_{g\in G} J(f,g) \leq \sup_{g\in G} \widehat{J}(f,g) + \mathfrak{R}_{F,G} + \left(\frac{M_z}{\sqrt{n}} + \frac{M_w}{\sqrt{\widetilde{n}}}\right)\sqrt{\log\frac{2}{\delta}}. \tag{6.10}$$

Moreover, recalling $W := \inf_{\boldsymbol{x}\in\mathcal{X}} |\mu_w(\boldsymbol{x})|$ and $\sup_{g\in L^2(p)} J(f,g) = \mathbf{E}[(\mu_w(\boldsymbol{x})f(\boldsymbol{x}) - \mu_z(\boldsymbol{x}))^2]$, we have

$$\begin{aligned}
\mathbf{E}\left[(f(\boldsymbol{x}) - u(\boldsymbol{x}))^2\right] &= \mathbf{E}\left[\left(f(\boldsymbol{x}) - \frac{\mu_z(\boldsymbol{x})}{\mu_w(\boldsymbol{x})}\right)^2\right] && (6.11)\\
&\leq \frac{1}{W^2}\,\mathbf{E}[(\mu_w(\boldsymbol{x})f(\boldsymbol{x}) - \mu_z(\boldsymbol{x}))^2] && (6.12)\\
&= \frac{1}{W^2}\left[\varepsilon_G(f) + \sup_{g\in G} J(f,g)\right]. && (6.13)
\end{aligned}$$

Combining Eq. (6.10) and Eq. (6.13) yields the inequality of the theorem. □

A.6 Proof of Corollary 5.5.1

Corollary 5.5.1. Let $F = \{x \mapsto \boldsymbol{\alpha}^\top \boldsymbol{\phi}(\boldsymbol{x}) \mid \|\boldsymbol{\alpha}\|_2 \leq \Lambda_F\}$, $G = \{x \mapsto \boldsymbol{\beta}^\top \boldsymbol{\psi}(\boldsymbol{x}) \mid \|\boldsymbol{\beta}\|_2 \leq \Lambda_G\}$, and assume that $r_F := \sup_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{\phi}(\boldsymbol{x})\|_2 < \infty$ and $r_G := \sup_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{\psi}(\boldsymbol{x})\|_2 < \infty$, where $\|\cdot\|_2$ is the L_2-norm. Under the assumptions of Theorem 5.5.1, it holds with probability at least $1-\delta$ that for every $f \in F$,

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[(f(\boldsymbol{x}) - u(\boldsymbol{x}))^2] \leq \frac{1}{W^2}\left[\sup_{g\in G}\widehat{J}(f,g) + \frac{C_z\sqrt{\log\frac{2}{\delta}} + D_z}{\sqrt{2n}} + \frac{C_w\sqrt{\log\frac{2}{\delta}} + D_w}{\sqrt{2\widetilde{n}}} + \varepsilon_G(f)\right],$$

where $C_z := r_G^2\Lambda_G^2 + 4r_G\Lambda_G M_{\mathcal{Y}}$, $C_w := 2r_F^2\Lambda_F^2 + 2r_F r_G\Lambda_F\Lambda_G + r_G^2\Lambda_G^2$, $D_z := r_G^2\Lambda_G^2/2 + 4r_G\Lambda_G M_{\mathcal{Y}}$, and $D_w := r_G^2\Lambda_G^2/2 + 4r_F r_G\Lambda_F\Lambda_G$.

Proof. Under the assumptions, it is known that the Rademacher complexity of the linear-in-parameter model F can be upper bounded as follows (Mohri et al., 2012):

$$\mathfrak{R}_p^N(F) \leq \frac{r_F\Lambda_F}{\sqrt{N}}.$$

We can bound $\mathfrak{R}_p^N(G)$ similarly. Applying these bounds to Theorem 5.5.1, we obtain the statement of Corollary 1. □

A.7 Proof of Theorem 5.5.2

We prove the following, formal version of Theorem 5.5.2.

Theorem 5.5.2. Under the assumptions of Corollary 5.5.1, it holds with probability at least $1-\delta$ that $\mathbf{E}[(\widehat{f}(\boldsymbol{x}) - u(\boldsymbol{x}))^2] \leq (4e_{n,\delta} + 2\varepsilon_G^F + \varepsilon_F)/W^2$, where $\varepsilon_G^F := \sup_{f\in F}\varepsilon_G(f)$, and

$\varepsilon_F := \inf_{f\in F} J(f)$, $\widehat{f} \in F$ is any approximate solution to $\inf_{f\in F}\sup_{g\in G}\widehat{J}(f,g)$ satisfying $\sup_{g\in G}\widehat{J}(\widehat{f},g) \leq \inf_{f\in F}\sup_{g\in G}\widehat{J}(f,g) + e_{n,\delta}$, and

$$e_{n,\delta} := \frac{C_z\sqrt{\log\frac{2}{\delta}} + D_z}{\sqrt{2n}} + \frac{C_w\sqrt{\log\frac{2}{\delta}} + D_w}{\sqrt{2\widetilde{n}}}.$$

Proof. Let $J(f) := \sup_{g\in L^2} J(f,g) = \mathbf{E}[(\mu_w(\boldsymbol{x})f(\boldsymbol{x}) - \mu_z(\boldsymbol{x}))^2]$, $J_G(f) := \sup_{g\in G} J(f,g)$, $\widehat{J}_G(f) := \sup_{g\in G}\widehat{J}(f,g)$. Let $\widetilde{f} \in F$ be any approximate solution to $\inf_{f\in F} J(f)$ satisfying $J(\widetilde{f}) \leq \varepsilon_F + e_{n,\delta}$.

As a special case of Eq. 6.10, we can prove that with probability at least $1-\delta$, it holds for every $f \in F$ that $J_G(f) \leq \widehat{J}_G(f) + e_{n,\delta}$. From Corollary 5.5.1, it holds that with probability at least $1-\delta$,

$$\begin{aligned}
J(\widehat{f}) &\leq \left[J(\widehat{f}) - J_G(\widehat{f})\right] + \left[J_G(\widehat{f}) - \widehat{J}_G(\widehat{f})\right] + \left[\widehat{J}_G(\widehat{f}) - \widehat{J}_G(\widetilde{f})\right] \\
&\quad + \left[\widehat{J}_G(\widetilde{f}) - J_G(\widetilde{f})\right] + \left[J_G(\widetilde{f}) - J(\widetilde{f})\right] + J(\widetilde{f}) \\
&\leq \varepsilon_G^F + e_{n,\delta} + e_{n,\delta} \\
&\quad + e_{n,\delta} + \varepsilon_G^F + [\varepsilon_F + e_{n,\delta}] \\
&\leq 4e_{n,\delta} + 2\varepsilon_G^F + \varepsilon_F.
\end{aligned}$$

Since $\mathbf{E}[(\widehat{f}(\boldsymbol{x}) - u(\boldsymbol{x}))^2] \leq \frac{1}{W^2} J(\widehat{f})$, we obtain the bound in Theorem 5.5.2. □

A.8 Binary Outcomes

When outcomes y take on binary values, e.g., success or failure, without loss of generality, we can assume that $y \in \{-1, 1\}$. Then, by the definition of the individual uplift, $u(\boldsymbol{x}) \in [-2, 2]$ for any $\boldsymbol{x} \in \mathbb{R}^d$. In order to incorporate this fact, we may add the following range constraints on f: $-2 \leq f(\boldsymbol{x}) \leq 2$ for every $\boldsymbol{x} \in \{\boldsymbol{x}_i\}_{i=1}^n \cup \{\widetilde{\boldsymbol{x}}_i\}_{i=1}^{\widetilde{n}}$.

A.9 Cases Where $p_1(\boldsymbol{x}) \neq p_2(\boldsymbol{x})$ or $(n_1, \widetilde{n}_1) \neq (n_1, \widetilde{n}_1)$

So far, we have assumed that $p_1(\boldsymbol{x}) = p_2(\boldsymbol{x})$, $m_1 = m_2$, and $n_1 = n_2$. The proposed method can be adapted to the more general case where these assumptions may not hold.

Let $r_k(\boldsymbol{x}) = \frac{n}{2n_k} \cdot \frac{p(\boldsymbol{x})}{p_k(\boldsymbol{x})}$ and $\widetilde{r}_k(\boldsymbol{x}) = \frac{\widetilde{n}}{2\widetilde{n}_k} \cdot \frac{p(\boldsymbol{x})}{p_k(\boldsymbol{x})}$, $k = 1, 2$, for every $\boldsymbol{x}$ with $p_k(\boldsymbol{x}) > 0$. Let $k_i := 1$ if the sample $\boldsymbol{x}_i$ originally comes from $p_1(\boldsymbol{x})$, and $k_i := 2$ if it comes from $p_2(\boldsymbol{x})$. Similarly, define $\widetilde{k}_i \in \{1, 2\}$ according to whether $\widetilde{\boldsymbol{x}}_i$ comes from $p_1(\boldsymbol{x})$ or $p_2(\boldsymbol{x})$. Then, unbiased estimators of the three terms in the proposed objective Eq. (5.10) are given as the

following weighted sample averages using r_k and $\widetilde{r}_k$:

$$\begin{aligned}
\mathop{\mathbf{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[wf(\boldsymbol{x})g(\boldsymbol{x})] &\approx \frac{1}{\widetilde{n}}\sum_{i=1}^{\widetilde{n}}[w_i f(\widetilde{\boldsymbol{x}}_i)g(\widetilde{\boldsymbol{x}}_i)\widetilde{r}_{\widetilde{k}_i}(\widetilde{\boldsymbol{x}}_i)], \\
\mathop{\mathbf{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[zg(\boldsymbol{x})] &\approx \frac{1}{n}\sum_{i=1}^{n}[z_i g(\boldsymbol{x}_i)r_{k_i}(\boldsymbol{x}_i)] \\
\mathop{\mathbf{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[g(\boldsymbol{x})^2] &\approx \frac{1}{2n}\sum_{i=1}^{n}[g(\boldsymbol{x}_i)^2 r_{k_i}(\boldsymbol{x}_i)] + \frac{1}{2\widetilde{n}}\sum_{i=1}^{\widetilde{n}}[g(\widetilde{\boldsymbol{x}}_i)^2\widetilde{r}_{\widetilde{k}_i}(\widetilde{\boldsymbol{x}}_i)].
\end{aligned}$$

The density ratios $p_k(\boldsymbol{x})/p(\boldsymbol{x})$ can be accurately estimated using i.i.d. samples from $p_k(\boldsymbol{x})$ and $p(\boldsymbol{x})$ (Liu et al., 2017; Nguyen et al., 2010; Sugiyama et al., 2012; Yamada et al., 2013).

A.10 Unbiasedness of the Weighted Sample Average

Below, we show that the weighted sample averages are unbiased estimates. We only prove for $\mathbf{E}[wf(\boldsymbol{x})g(\boldsymbol{x})]$ since the other cases can be proven similarly. The expectation of the weighted sample average transforms as follows:

$$\begin{aligned}
&\frac{1}{\widetilde{n}}\sum_{i=1}^{\widetilde{n}}\mathop{\mathbf{E}}_{\widetilde{\boldsymbol{x}}_i^{(k)}\sim p_k(\boldsymbol{x}),t_i^{(k)}\sim p_k(t|\widetilde{\boldsymbol{x}}_i^{(k)})}\left[w_i f(\widetilde{\boldsymbol{x}}_i)g(\widetilde{\boldsymbol{x}}_i)\widetilde{r}_{\widetilde{k}_i}(\widetilde{\boldsymbol{x}}_i)\right] \\
&= \frac{1}{\widetilde{n}}\sum_{k=1,2}\sum_{i=1}^{\widetilde{n}_k}\mathop{\mathbf{E}}_{\boldsymbol{x}\sim p_k(\boldsymbol{x}),t\sim p_k(t|\boldsymbol{x})}\left[(-1)^{k-1}tf(\boldsymbol{x})g(\boldsymbol{x})\frac{\widetilde{n}}{2\widetilde{n}_k}\cdot\frac{p(\boldsymbol{x})}{p_k(\boldsymbol{x})}\right] \\
&= \frac{1}{2}\sum_{k=1,2}\mathop{\mathbf{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x}),t\sim p_k(t|\boldsymbol{x})}\left[(-1)^{k-1}tf(\boldsymbol{x})g(\boldsymbol{x})\right] \\
&= \iint(-1)^{k-1}t\sum_{k=1,2}\frac{1}{2}p_k(t\mid\boldsymbol{x})f(\boldsymbol{x})g(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}t\mathrm{d}\boldsymbol{x} \\
&= \iint wp(w\mid\boldsymbol{x})f(\boldsymbol{x})g(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}t\mathrm{d}\boldsymbol{x} \\
&= \mathop{\mathbf{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x}),w\sim p(w|\boldsymbol{x})}[wf(\boldsymbol{x})g(\boldsymbol{x})].
\end{aligned}$$

A.11 Gaussian Basis Functions Used in Experiments

The l-th element of $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}),\ldots,\phi_{b_f}(\boldsymbol{x}))^\top$ is defined by

$$\phi_l(\boldsymbol{x}) := \exp\left(\frac{-\|\boldsymbol{x}-\boldsymbol{x}^{(l)}\|^2}{\sigma^2}\right),$$

where $\boldsymbol{x}^{(l)}$, $l=1,\ldots,b_f$, are randomly chosen training data points. We used $b_f = 100$ and $\sigma = 25$ for all experiments. $\boldsymbol{\psi}$ is defined similarly.

A.12 Justification of the Sub-Sampling Procedure

Suppose that we want a sample subset S_k following the treatment policy $p_k(t \mid \boldsymbol{x})$. For each sample $(\boldsymbol{x}_i, t_i, y_i) \sim p(\boldsymbol{x}, t, y)$ in the original dataset, we randomly add it into S_k with probability proportional to $p_k(t_i \mid \boldsymbol{x}_i)/p(t_i \mid \boldsymbol{x}_i)$. Then,

$$\begin{aligned} p(\boldsymbol{x}_i, t_i, y_i \mid (\boldsymbol{x}_i, t_i, y_i) \in S_k) &= \frac{p((\boldsymbol{x}_i, t_i, y_i) \in S_k \mid \boldsymbol{x}_i, t_i, y_i) p(\boldsymbol{x}_i, t_i, y_i)}{\int \sum_{y_i, t_i} p((\boldsymbol{x}_i, t_i, y_i) \in S_k \mid \boldsymbol{x}_i, t_i, y_i) p(\boldsymbol{x}_i, t_i, y_i) \mathrm{d}\boldsymbol{x}_i} \\ &= \frac{p_k(t_i \mid \boldsymbol{x}_i) p(y_i \mid \boldsymbol{x}_i, t_i) p(\boldsymbol{x}_i)}{\int \sum_{y_i, t_i} p_k(t_i \mid \boldsymbol{x}_i) p(y_i \mid \boldsymbol{x}_i, t_i) p(\boldsymbol{x}_i) \mathrm{d}\boldsymbol{x}_i} \\ &= p_k(t_i \mid \boldsymbol{x}_i) p(y_i \mid \boldsymbol{x}_i, t_i) p(\boldsymbol{x}_i). \end{aligned}$$

This means that the subsamples S_k preserve the original $p(y \mid \boldsymbol{x}, t)$ and $p(\boldsymbol{x})$ but follow the desired treatment policy $p_k(t \mid \boldsymbol{x})$.

A.13 McDiarmid's Inequality

Although McDiarmid's inequality is a well known theorem, we present the statement to make the dissertation self-contained.

Theorem A.1 (McDiarmid's inequality)**.** *Let $\varphi : \mathcal{D}^N \to \mathbb{R}$ be a measurable function. Assume that there exists a real number $B_\varphi > 0$ such that*

$$|\varphi(v_1, \ldots, v_N) - \varphi(v'_1, \ldots, v'_N)| \leq B_\varphi, \tag{6.14}$$

for any $v_i, \ldots, v_N, v_1, \ldots, v'_N \in \mathcal{D}$ where $v_i = v'_i$ for all but one $i \in \{1, \ldots, N\}$. Then, for any $\mathcal{D}$-valued independent random variables $V_1, \ldots, V_N$ and any $\delta > 0$ the following holds with probability at least $1 - \delta$:

$$\varphi(V_1, \ldots, V_N) \leq \mathbf{E}[\varphi(V_1, \ldots, V_N)] + \sqrt{\frac{B_\varphi^2 N}{2} \log \frac{1}{\delta}}.$$

B Supplementary Material for Multi-Task Principal Component Analysis

This section is for supplementary materials for Chapter 4.

B.1 Study of the proposed regularization

We highlight the behavior of the proposed regularization on a illustrative example. Let us define the following orthogonal skinny matrices $\boldsymbol{U}, \boldsymbol{U}' \in \mathbb{R}^{d \times 2}$ such that $\boldsymbol{U} = \left[\begin{array}{c|c} \boldsymbol{u} & \boldsymbol{v} \end{array}\right]$ and $\boldsymbol{U}' = \left[\begin{array}{c|c} \boldsymbol{v} & \boldsymbol{u} \end{array}\right]$, with $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$ such that $\boldsymbol{u}^\top \boldsymbol{u} = 1$, $\boldsymbol{v}^\top \boldsymbol{v} = 1$ and $\boldsymbol{u}^\top \boldsymbol{v} = 0$.

Even if $\boldsymbol{U}$ and $\boldsymbol{U}'$ span the same subspace of $\mathbb{R}^d$, we show that the proposed regularization is more interesting than the naive one. Indeed, the spanned subspaces being the same, we seek a regularization giving a high result for $\boldsymbol{U}$ and $\boldsymbol{U}'$.

The scalar product regularization (i.e. the naive regularization) gives the following[*1]:

$$\begin{aligned}
\mathrm{Tr}(\boldsymbol{U}^\top \boldsymbol{U}') &= \mathrm{Tr}\left(\left[\begin{array}{c|c} \boldsymbol{u} & \boldsymbol{v} \end{array}\right]^\top \left[\begin{array}{c|c} \boldsymbol{v} & \boldsymbol{u} \end{array}\right]\right) \\
&= \mathrm{Tr}\left(\begin{bmatrix} \boldsymbol{u}^\top \boldsymbol{v} & \boldsymbol{u}^\top \boldsymbol{u} \\ \boldsymbol{v}^\top \boldsymbol{v} & \boldsymbol{v}^\top \boldsymbol{u} \end{bmatrix}\right) \\
&= \mathrm{Tr}\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right) \\
&= 0
\end{aligned}$$

On the contrary, our proposed regularization gives the following:

$$\begin{aligned}
\mathrm{Tr}(\boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{U}'\boldsymbol{U}'^\top) &= \mathrm{Tr}(\boldsymbol{U}'^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{U}') \\
&= \mathrm{Tr}\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right) \\
&= \mathrm{Tr}\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \\
&= 2
\end{aligned}$$

Hence, contrary naive regularization, our regularization is robust to changes in the order of the basis vectors describing the span of subspace. This example is the simplest case as our regularization is invariant to orthogonal transformations (which includes basis re-ordering).

B.2 Additional numerical experiments

In this section, we provide some numerical results in a slightly different setup. This experimental setup is designed to study a practical application of MTL approaches, where data are available for several tasks and a new related task to be solved appears. Instead of using only the few available data for this new task, we use the MTL methodology to transfer knowledge from the already-known tasks to the new one. We refer to this as the *adaptation setup* and study it on both synthetic and BCI data.

B.2.1 Adaptation Setup

In the adaptation setup, we pick one task as the target task and draw a small number of training samples for the task but a (relatively) large number of samples for the other tasks. We run the proposed method to solve all the tasks as in the scarce setup, but we throw away the obtained subspaces except that for the target task.

[*1] Note however that this $\mathrm{Tr}(\boldsymbol{U}^\top \boldsymbol{U}) = 2$.

Table B.1: The averages and the standard errors of the RVRs over 100 trials of experiments on the toy dataset in the adaptation setup. The best and comparable to the best scores are shown in bold face.

	RMT-PCA	I-PCA	C-PCA
$k = 1$	0.2374(0.0005)	0.2146(0.0005)	**0.2499(0.0001)**
$k = 2$	**0.4266(0.0010)**	0.4087(0.0007)	0.4160(0.0010)
$k = 3$	**0.5938(0.0010)**	0.5816(0.0009)	0.5836(0.0009)
$k = 4$	**0.7574(0.0009)**	0.7353(0.0008)	0.7503(0.0009)
$k = 5$	0.9024(0.0007)	0.8731(0.0006)	**0.9166(0.0000)**

For evaluation, we use RVR for evaluation as in the scarce setup, but we only evaluate that for the target task. In other words, we assess the performance with the following score r:

$$r = \frac{\mathrm{Tr}(\widehat{\boldsymbol{U}}_\tau^\top \widehat{\boldsymbol{C}}'_\tau \widehat{\boldsymbol{U}}_\tau)}{\mathrm{Tr}(\widehat{\boldsymbol{C}}'_\tau)}, \tag{6.15}$$

where task τ is the target task, $\widehat{\boldsymbol{U}}_\tau$ denotes an arbitrary orthogonal basis matrix of the estimated subspace, $\widehat{\boldsymbol{C}}'_\tau$ is the sample covariance matrix calculated using test samples. In regularization parameter selection by cross-validation, we also use this score but calculated with hold-out samples in place of the test samples.

As in the scarce setup, the training sample size and the test sample size differ from one dataset to another. See Section 4.3.2 for the specific numbers. Also, we run several trials of this experiment with different data realizations. See Section 4.3.2 for the specific numbers of the trials.

B.2.2 Results

We show the results for the adaptation setup below.

Performance Transition over Regularization-Level Change The results on the toy data in the adaptation setup are summarized in Figure B.1. We observe similar behaviors as in the scarce setup (see Figure B.1).

The results on the BCI data in the adaptation setup are shown in Figure B.2. Similarly to the case of the toy data, the performance was improved for all the k with appropriate λ values.

These results show that the proposed method is useful in both the scarce and the adaptation setup as long as the regularization level is in a moderate range.

Regularization Parameter Selection by Cross-Validation Next, we apply a cross-validation procedure for selecting the regularization parameter on BCI data (in the adaptation setup). The results are summarized in Table B.2, showing that the proposed method with the cross-validated regularization parameter significantly outperforms the competitors.

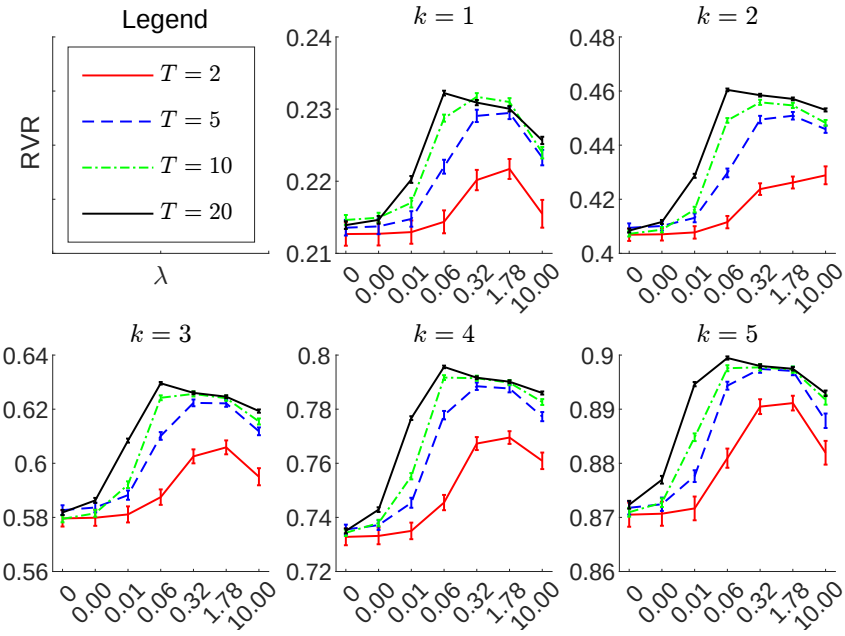
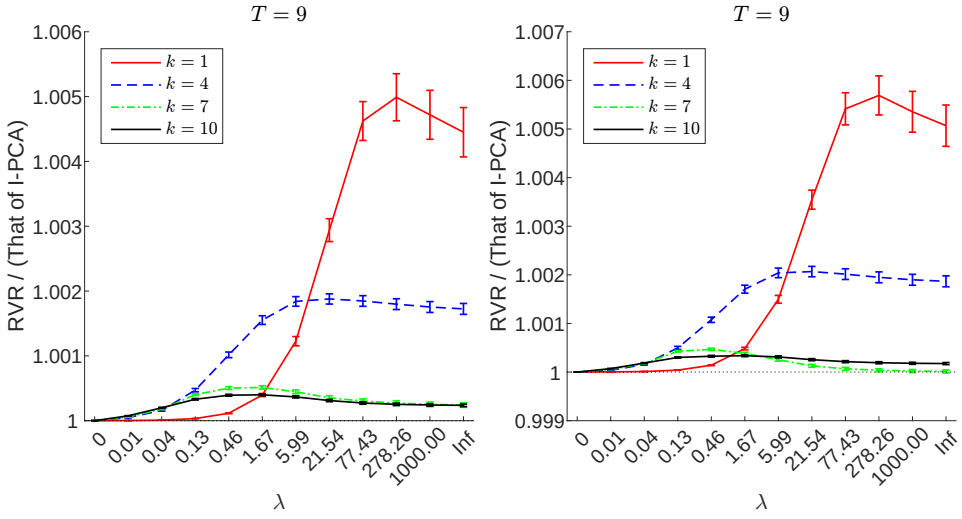


Figure B.1: The transition of the RVR score over the level of regularization on synthetic data. Each plot corresponds to a different dimensionality k, and each curve corresponds to a different number of tasks T. 'Inf' denotes infinity. The error bars show the mean scores and their standard errors over 100 trials of the experiment.



(A) Class 1. (B) Class 2.

Figure B.2: Transition of the RVR score of the proposed method divided by the score of I-PCA over the the level of regularization on BCI data ('Inf' denotes infinity). Each plot corresponds to a different class, and each curve corresponds to a different dimensionality k. The black dotted lines indicate ratio of 1. The error bars show the mean scores and their standard errors over 72 trials.

Table B.2: The averages and standard errors of the RVRs on BCI data in the adaptation setup. In bold face are shown the best and comparable to the best scores by a paired t-test (5% significance level).

	Multitask	Independent	Common
(Class 1)			
$k = 1$	**0.8067(0.0001)**	0.8055(0.0002)	0.8053(0.0001)
$k = 4$	**0.9687(0.0000)**	0.9682(0.0001)	0.9676(0.0000)
$k = 7$	**0.9884(0.0000)**	0.9882(0.0000)	0.9871(0.0000)
$k = 10$	**0.9949(0.0000)**	0.9948(0.0000)	0.9941(0.0000)
(Class 2)			
$k = 1$	**0.7898(0.0002)**	0.7884(0.0003)	0.7881(0.0001)
$k = 4$	**0.9668(0.0001)**	0.9662(0.0001)	0.9654(0.0000)
$k = 7$	**0.9877(0.0000)**	0.9875(0.0000)	0.9861(0.0000)
$k = 10$	**0.9946(0.0000)**	0.9945(0.0000)	0.9939(0.0000)

List of Figures

List of Tables